Paolo Camagni
Riccardo Nikolassy

DATABASE SQL & PHP

Per il quinto anno degli Istituti Tecnici Tecnologici

Con prove per la nuova maturità

Edizione **OPENSCHOOL**

1	**LIBRO**DITESTO
2	**E-BOOK+**
3	RISORSE**ONLINE**
4	**PIATTAFORMA**

HOEPLI

PAOLO CAMAGNI RICCARDO NIKOLASSY

Database SQL & php

Per il quinto anno degli Istituti Tecnici Tecnologici
Con prove per la nuova maturità

EDITORE ULRICO HOEPLI MILANO

Indice

Unita I
Progetto di database

Area digitale

Unita 2
Database Management System (DBMS)

Area digitale

Unita 3
Il linguaggio SQL

Area digitale

Unita 4
Programmazione lato server con php

Area digitale

- Conversioni implicite in php
- Classe COM non presente
- La funzione phpinfo()
- Richiami sul ruolo del server HTTP

- Esercizi
- Esercizi per l'approfondimento

Unita 5
La prova scritta di Informatica

Puoi scaricare le Lezioni 4, 5 anche da hoepliscuola.it

Presentazione

Database SQL & php è un nuovo volume destinato al quinto anno del corso di Informatica degli Istituti Tecnici settore Tecnologico indirizzo Informatica e Telecomunicazioni.

L'opera, concepita secondo le recenti indicazioni ministeriali, tiene conto anche delle indicazioni ricevute dai docenti che hanno in uso la precedente edizione: in particolare si è preso come riferimento ispiratore le seconde prove di maturità proposte dal Ministero negli ultimi anni, dalle quali si è preso spunto per organizzare i contenuti in due macroaree:
– il progetto di database e i DBMS;
– la progettazione di pagine dinamiche in php e l'interrogazione dei database in linguaggio SQL.

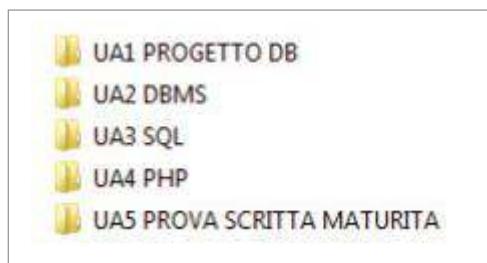
Nel **progetto di database** si è dato ampio spazio agli esempi, proposti con difficoltà graduale, presentando la fase di modellazione concettuale sia con l'utilizzo di schemi con la notazione grafica proposta da **Chen** (diagramma E-R) sia con schemi realizzati secondo la grafica **UML**.
La gestione delle basi di dati mediante DBMS viene esposta secondo la filosofia stand-alone in **Access** e secondo la filosofia distribuita mediante **MySQL**.
Vengono affrontate le tecniche di progettazione delle query in linguaggio **SQL**, sia per la creazione di tabelle sia per la realizzazione di interrogazioni anche complesse. Un'ampia sezione riguarda il linguaggio **php**, che consente di realizzare applicazioni **client-server**, secondo il modello three-tier.

STRUTTURA DEL TESTO, METODOLOGIA E STRUMENTI DIDATTICI

Il testo, suddiviso in **cinque Unità di apprendimento**, ciascuna composta da più lezioni, sviluppa i seguenti argomenti:
– **progetto di database**, **Access** e linguaggio **SQL**;
– sviluppo di siti Web dinamici aziendali con **php**;
– **prova scritta** di informatica (soluzione completa delle prove d'esame proposte da "La buona scuola").

Le finalità e i contenuti dei diversi argomenti affrontati sono descritti negli **obiettivi disciplinari** e nelle indicazioni *In questa lezione impareremo*. Alla fine di ogni lezione sono presenti **esercizi, anche interattivi, di valutazione delle conoscenze** e **delle competenze** raggiunte.
Ogni unità si conclude con una sezione utile per la simulazione operativa delle attività previste nell'**Alternanza Scuola-Lavoro** e con una scheda **CLIL**.

ESPANSIONI DIGITALI

La nuova edizione Openschool consente di:
– scaricare gratuitamente il libro digitale arricchito (eBook+); l'eBook+ permette in particolare di:
  • eseguire tutte le esercitazioni a risposta chiusa in modo interattivo;
  • scaricare gli approfondimenti tematici e gli esercizi per l'approfondimento;
  • scaricare le lezioni integrative;
– disporre di ulteriori esercitazioni online, utilizzabili a discrezione del docente per classi virtuali gestibili attraverso la piattaforma Open.

RISORSE ONLINE E PIATTAFORMA DIDATTICA

Sul sito www.hoepliscuola.it (hoepliscuola.it) sono disponibili **numerosi materiali**. In particolare, per lo studente: **approfondimenti**, **esercizi** di recupero, rinforzo e approfondimento.
Inoltre, i **file** richiamati nelle lezioni e nelle esercitazioni contenuti nel CD-ROM allegato al volume sono scaricabili anche dal sito.

CD-ROM

Il CD-ROM allegato al volume contiene i file degli esempi, degli esercizi e di eventuali soluzioni, nonché il materiale necessario per eseguire le procedure guidate passo passo.

Inoltre i codici delle soluzioni dei temi di maturità descritti nella **UDA 5** e di altri particolarmente significativi sono disponibili nella cartella SOLUZIONI MATURITA: è sufficiente copiarne il contenuto nella cartella locale di **XAMPP**, cioè in **C:\xampp\htdocs** creando la sottocartella **maturita**.

Struttura del corso per immagini

APERTURA UNITÀ

L'unità si apre con l'indice delle lezioni sviluppate e l'indicazione degli obiettivi generali suddivisi in conoscenze, competenze e abilità.

APERTURA LEZIONE

La lezione si apre con una breve sintesi degli argomenti trattati e con la schematizzazione dei contenuti attraverso una MAPPA CONCETTUALE.

EVIDENZIAZIONE

Connota dei concetti da ricordare.

WIKI

Specifica il significato di un termine.

ATTENZIONE

Individua aspetti su cui focalizzare l'attenzione.

ESEMPIO

Gli esempi chiariscono i concetti appena esposti e svolgono la funzione di traccia di svolgimento per lo studente.

PER SAPERNE DI PIÙ

Schede di approfondimento degli argomenti sviluppati nel volume.

METTITI ALLA PROVA

Appendice esercitativa che prende spunto dal problema di partenza accrescendone le funzionalità e il campo applicativo.

VERIFICA LE CONOSCENZE/COMPETENZE

Presenti a fine lezione, propongono varie tipologie di test (quelli a risposta a chiusa sono interattivi e autocorrettivi) e di esercizi, con l'indicazione dei problemi connessi ai compiti di realtà.

VERIFICA LE COMPETENZE

Presente al termine di ogni unità, contiene esercizi sommativi per la verifica dei saperi essenziali, con indicati i problemi connessi ai compiti di realtà.

UNITÀ 2 - Database Management System (DBMS)

ALTERNANZA SCUOLA-LAVORO

Proposta operativa n. 1

GESTIONE MERCI SUDDIVISE IN SEDI DIVERSE

PROBLEMA
Il tuo tutor aziendale ti chiede di progettare un database per la gestione dei magazzini di un'azienda con più sedi. Le merci sono conservate in magazzini, e in ciascuno di essi è presente una sola categoria merceologica.

Passi operativi
– Crea la struttura della base di dati.
– Individua la struttura delle tabelle coinvolte.
– Crea una maschera per l'immissione dei dati nelle tabelle.
– Crea una maschera con sottomaschera per la visualizzazione dei prodotti per magazzino.
– Definisci le seguenti interrogazioni per ottenere:
  • un elenco di eventuali prodotti presenti in più di un magazzino;
  • un elenco degli articoli che hanno subito operazioni di scarico in un determinato mese;
  • l'elenco dei movimenti di ciascun articolo nell'ultimo anno.
– Crea un prospetto dei prodotti suddivisi per magazzino in grado di calcolare alcuni dati statistici (media, somma, conteggio, minimo ecc.).

Proposta operativa n. 2

STUDIO DI AMMINISTRAZIONE CONDOMINIALE

PROBLEMA
Il tuo tutor aziendale, che dirige uno studio di amministrazione di condomini, ti chiede di realizzare una semplice procedura per la gestione di alcuni dati relativi ai fornitori e alle prestazioni a loro connesse (acquisti di materiale, solleciti, gestione raccomandate, richieste di riparazioni urgenti, ecc.). Lo scopo è quello di avviare un processo di informatizzazione dello studio per produrre le stampe della situazione contabile di ciascun condominio e fornitore. Per realizzare il database dovrai collaborare con la segretaria dello studio che potrà fornirti tutti i dati necessari.

Passi operativi
– Crea la struttura della base di dati.
– Individua la struttura delle tabelle coinvolte.
– Crea una maschera per l'immissione dei dati nelle tabelle.
– Crea una maschera con sottomaschera per la visualizzazione dei fornitori e delle relative riparazioni effettuate.
– Definisci le seguenti interrogazioni per ottenere:
  • un elenco di fornitori che devono ancora essere retribuiti;
  • un elenco dei fornitori che non hanno mai ricevuto solleciti di consegna;
  • l'elenco del materiale acquistato nell'ultimo mese.
– Crea un prospetto del materiale acquistato dai diversi fornitori, suddiviso per mese, in grado di calcolare alcuni dati statistici (media, somma, conteggio, minimo ecc.).

207

ALTERNANZA SCUOLA-LAVORO

Presente al termine di ogni unità, contiene proposte di attività connesse con le esperienze di Alternanza Scuola-Lavoro.

PROVE SCRITTE DI MATURITÀ

Esempi di prove scritte di informatica con le relative soluzioni.

CLIL

Chiude ogni unità una scheda CLIL, che propone in inglese, i concetti chiave dell'unità e alcuni quesiti di diverse tipologie.

L'OFFERTA DIDATTICA **HOEPLI**

L'edizione **Openschool** Hoepli offre a docenti e studenti tutte le potenzialità di Openschool Network (ON), il nuovo sistema integrato di contenuti e servizi per l'apprendimento.

Edizione **OPENSCHOOL**

LIBRO DI TESTO

Il libro di testo è l'**elemento cardine** dell'offerta formativa, uno strumento didattico **agile** e **completo**, utilizzabile **autonomamente** o in combinazione con il ricco **corredo digitale** offline e online. Secondo le più recenti indicazioni ministeriali, volume cartaceo e apparati digitali **sono integrati in un unico percorso didattico.** Le espansioni accessibili attraverso l'eBook+ e i materiali integrativi disponibili nel sito dell'editore sono puntualmente richiamati nel testo tramite apposite icone.

eBOOK+

L'**eBook+** è la versione digitale e interattiva del libro di testo, utilizzabile su **tablet, LIM e computer**. Aiuta a comprendere e ad approfondire i contenuti, rendendo l'apprendimento più attivo e coinvolgente. Consente di leggere, annotare, sottolineare, effettuare ricerche e accedere direttamente alle numerose **risorse digitali integrative**.

➜ Scaricare l'eBook+ è molto **semplice**. È sufficiente seguire le istruzioni riportate nell'ultima pagina di questo volume.

RISORSE ONLINE

Il sito della casa editrice offre una ricca dotazione di **risorse digitali** per l'approfondimento e l'aggiornamento. Nella pagina web dedicata al testo è disponibile **MyBookBox**, il contenitore virtuale che raccoglie i materiali integrativi che accompagnano l'opera.

➜ Per accedere ai materiali è sufficiente registrarsi al sito **www.hoepliscuola.it** e inserire il codice coupon che si trova nella terza pagina di copertina. **Per il docente** nel sito sono previste ulteriori risorse didattiche dedicate.

PIATTAFORMA DIDATTICA

La **piattaforma didattica** è un ambiente digitale che può essere utilizzato in modo duttile, a misura delle esigenze della classe e degli studenti. Permette in particolare di **condividere contenuti** ed **esercizi** e di partecipare a **classi virtuali**. Ogni attività svolta viene salvata sul **cloud** e rimane sempre disponibile e aggiornata. La piattaforma consente inoltre di consultare la versione online degli eBook+ presenti nella propria libreria.

➜ È possibile accedere alla piattaforma attraverso il sito **www.hoepliscuola.it**.

Progetto di database

UNITÀ
CONOSCENZE

- Comprendere l'utilità dei database
- Conoscere i vantaggi di un DBMS
- Acquisire la conoscenza degli aspetti funzionali e organizzativi di una base di dati
- Conoscere il concetto di dipendenza funzionale
- Comprendere le motivazioni alla base della normalizzazione

COMPETENZE

- Utilizzare lo schema concettuale dei dati E-R
- Individuare le entità e le relazioni tra le entità all'interno di una situazione complessa
- Utilizzare il modello logico dei dati
- Utilizzare gli operatori relazionali
- Rispettare le regole di integrità

ABILITÀ

- Utilizzare modelli per descrivere processi aziendali
- Applicare le gerarchie di generalizzazione
- Utilizzare le potenzialità di una base di dati relazionale
- Applicare le regole di normalizzazione
- Progettare basi di dati relazionali

Il file relativi a esempi, esercizi e a eventuali soluzioni di questa unità sono contenuti nel CD-ROM e sono anche scaricabili dal sito hoepliscuola.it

Introduzione ai database

IN QUESTA LEZIONE IMPAREREMO...

- quali sono i limiti degli archivi classici
- in che cosa consiste un DBMS
- quali sono le caratteristiche e le prestazioni di un DBMS

MAPPA CONCETTUALE

Generalità

In ogni applicazione informatica, dalla più complessa alla più semplice, vengono trattate informazioni ed è necessario che queste informazioni vengano memorizzate in modo permanente per essere poi utilizzate in successive elaborazioni.

La memorizzazione delle informazioni in file di record mediante un linguaggio di programmazione è un'operazione complessa, laboriosa e pesante da un punto di vista computazionale; inoltre le tecniche classiche di organizzazione degli archivi su memoria di massa non sempre corrispondono adeguatamente alle strutture dei dati che devono essere rappresentati, che aumentano di complessità con l'aumento delle dimensioni della situazione che deve essere gestita.

Si pensi alle differenze dimensionali e di complessità tra la gestione dei dati archiviati in una agenda telefonica personale di un telefono cellulare e quelli di una contabilità aziendale.

Nelle aziende, in generale, le esigenze e le necessità di memorizzare ed elaborare le informazioni tra loro totalmente disomogenee sono sempre molteplici: si pensi alla gestione degli stipendi, alla pianificazione della produzione, alle contabilità, al magazzino ecc.

Ogni programma (funzionalità) avrà pertanto la necessità di avere una propria organizzazione dei dati che solitamente risulta di difficile modificabilità e adattabilità ad altre situazioni e attività.
Per affrontare in modo semplice, organico e flessibile tutte le situazioni che quotidianamente si presentano nelle aziende occorre utilizzare strumenti più potenti per gestire le informazioni: questi strumenti sono i database.
Attraverso i database è possibile memorizzare e gestire in modo flessibile ed efficiente le informazioni che sono "il vero patrimonio di ogni organizzazione".

Ogni applicazione software presente in ciascuna organizzazione deve quindi poter facilmente:
- ricercare e recuperare le informazioni in base a determinati criteri di ricerca;
- selezionarle e raggrupparle secondo esigenze operative;
- aggiungerne di nuove;
- modificarle aggiornandone il valore nel tempo;
- cancellare quelle che non hanno più contenuto informativo utile.

ESEMPIO

Per avere un'idea di che cosa s'intende per informazione che deve essere elaborata basti pensare ai dati che devono essere gestiti dall'ufficio anagrafe di un Comune, oppure alla contabilità di una grande azienda, alla gestione delle vendite o alla gestione dei conti correnti di una banca, alle prenotazioni di un albergo, di un'agenzia turistica o di un aeroporto.

Da quanto finora detto possiamo formulare pertanto una prima definizione di base di dati.

Una base di dati o database è una raccolta di dati progettati in modo tale da poter essere utilizzati in maniera ottimizzata da differenti applicazioni e da utenti diversi.

In questa lezione presenteremo la teoria comune a tutti i database e descriveremo i principi fondamentali che ne guidano la creazione e la gestione, mentre nella successiva verrà fornita la classificazione delle diverse tipologie di database esistenti ed esaminato in maniera più approfondita il modello relazionale che, tra tutti i tipi di database, è sicuramente quello più utilizzato oggi in tutti gli ambienti di sviluppo software e che ha il grosso vantaggio di basarsi su una teoria matematica ben nota e consolidata, largamente conosciuta e utilizzata in tutto il mondo.

Indipendentemente dal produttore del database, che sia Oracle oppure Microsoft, la teoria che sta alla base del suo funzionamento è la stessa e quindi, una volta acquisita, permette di utilizzare qualsiasi tipo di database relazionale.

Necessità dei database

Abbiamo detto che le informazioni necessarie a un'organizzazione sono gestite da un sistema informativo.

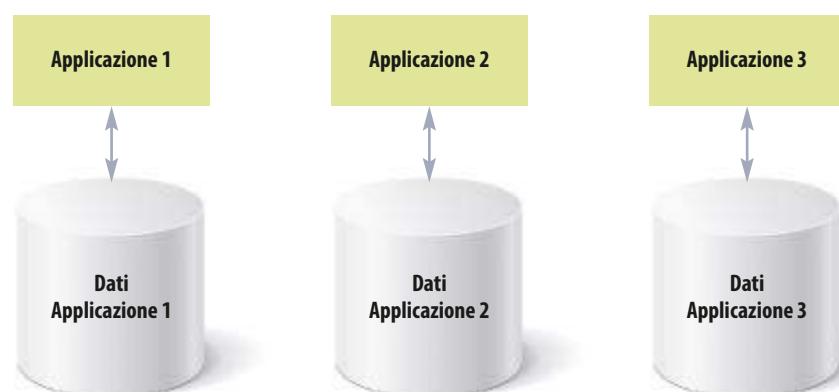
Un sistema informativo è un insieme organizzato di strumenti automatici, procedure manuali, risorse umane e materiali, norme organizzative, orientato alla gestione delle informazioni rilevanti per un'organizzazione.

Quindi, ogni informazione e in qualunque formato essa sia fa parte del sistema informativo aziendale, di cui il sistema informatico è una componente essenziale.

Un sistema informatico (spesso chiamato EDP, *Electronic Data Processing*) è un sottoinsieme del sistema informativo che si dedica alla gestione automatica delle informazioni, rappresentate mediante dati digitali.

In prima analisi, il sistema informatico (SI) è costituito dagli archivi elettronici in cui sono memorizzati tutti i dati relativi all'azienda, e cioè:
– i supporti fisici per la memorizzazione dei dati;
– le procedure di interrogazione per la ricerca delle informazioni (applicazioni);
– gli strumenti di comunicazione tra i terminali degli operatori.

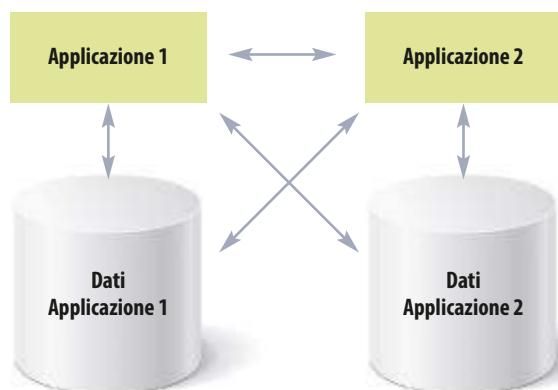
In base alla loro natura, possiamo individuare due componenti nel SI:
– archivi e applicazioni: componente software;
– supporti fisici e strumentazione: componente hardware.

Noi ci occuperemo della componente software.

Archivi e applicazioni informatiche

Passiamo innanzitutto a esaminare più nel dettaglio il significato di archivio e applicazione informatica.

L'archivio è l'insieme dei dati che vengono salvati su un supporto di memorizzazione.
L'applicazione informatica è invece la componente del sistema informatico che utilizza dati in esso immagazzinati per svolgere una funzione specifica all'interno dell'organizzazione di cui il SI fa parte.

Prendiamo ora in considerazione una generica azienda che non utilizzi i database e dove siano presenti diverse applicazioni informatizzate: lo schema di elaborazione tradizionale è rappresentato in figura, dove dati e applicazioni si presentano strutturati in modo autonomo e disgiunto.

Ogni singola applicazione opera su un insieme di dati memorizzati secondo una struttura definita all'interno dell'applicazione stessa dall'analista e dal programmatore e viene realizzata in modo da rendere massima la sua efficienza, indipendentemente dalle presenza o meno delle altre applicazioni.

In un sistema di questo tipo ogni applicazione opera dunque in maniera del tutto indipendente o quasi dalle altre applicazioni, facendo uso dei propri dati e dei propri programmi.

Le varie applicazioni risultano essere isolate le une dalle altre anche se fanno parte dello stesso sistema: in particolare, questo comporta dei problemi quando è necessaria la condivisione di dati da parte di due o più applicazioni, oppure quando un'applicazione utilizza i dati forniti da un'altra applicazione, come per esempio la fatturazione e la contabilità.

ESEMPIO

Un "classico" problema si verifica nel caso frequente in cui due applicazioni differenti lavorano utilizzando i dati da uno stesso file e una di esse ha bisogno di apportare delle modifiche alla struttura dei dati (per esempio deve aggiungere un nuovo campo in un record): in tal caso sarà necessario modificare anche la descrizione dei dati interna all'altra applicazione, anche se quest'ultima non utilizza il campo che è stato aggiunto.

A volte l'operazione di modifica contemporanea di tutte le applicazioni che utilizzano uno stesso file non è sempre facilmente eseguibile e capita spesso di dover arrivare a duplicare i file, con conseguenze facilmente intuibili. Si può per esempio arrivare al caso estremo in cui gli stessi dati vengono ripetuti tante volte quante sono le applicazioni che devono usarli, con conseguente spreco di memoria e con il rischio di introdurre inconsistenze ed errori.

Inoltre, l'accesso ai dati avviene solo tramite le applicazioni che li hanno generati, cioè tramite programmi specializzati, limitando di conseguenza le possibilità di estrarre ulteriori informazioni dagli stessi mediante elaborazioni autonome: per ottenere risultati diversi da quelli previsti dall'applicazione è necessario realizzare nuove procedure specifiche per le nuove richieste con notevole dispendio di tempo e denaro.

Dati, archivi e database

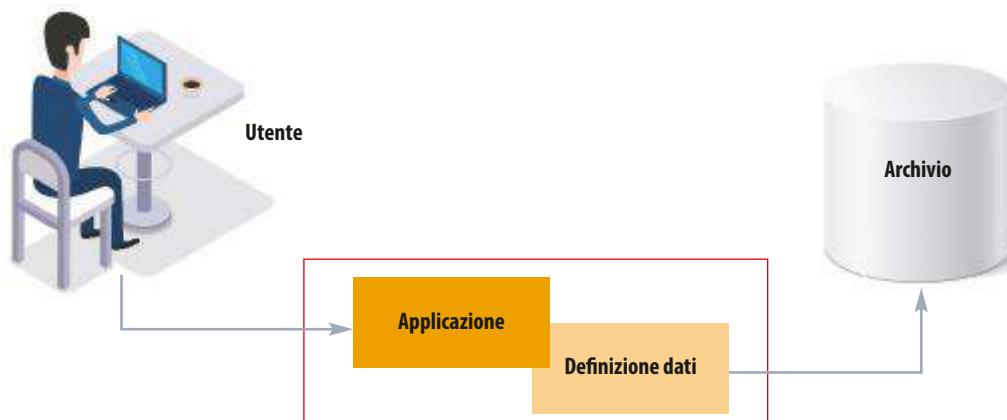
Per risolvere i problemi appena descritti è dunque necessario disporre di un sistema in grado di "contenere" tutti i dati necessari alle diverse applicazioni, in modo da averne una sola copia, unica e sempre aggiornata, disponibile a tutti i programmi, che consenta inoltre l'accesso simultaneo (concorrente) a più utenti, anche con applicazioni diverse e scritte in linguaggi di programmazione diversi.
Questo sistema deve poi essere in grado di offrire "una interfaccia" molto semplice che permetta di interagire con i dati e mettere a disposizione uno specifico linguaggio per rendere possibili le

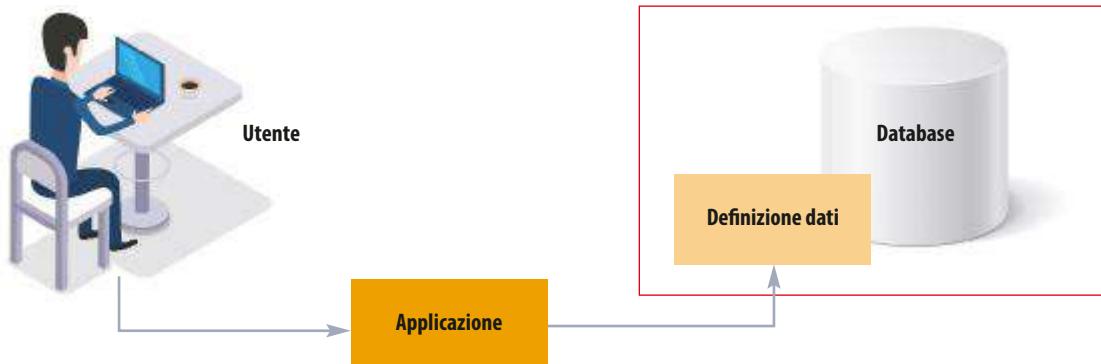
operazioni di archiviazione, modifica e recupero dei dati stessi. Infine, deve poter permettere di modificare la struttura dei dati senza che le applicazioni “già funzionanti” ne risentano in qualche modo, cioè “non se ne accorgano neppure”.
Tale strumento è proprio il database, composto dall’insieme degli archivi (tabelle) di tutte le applicazioni aziendali, che raccoglie univocamente tutti i dati, li organizza e li gestisce.

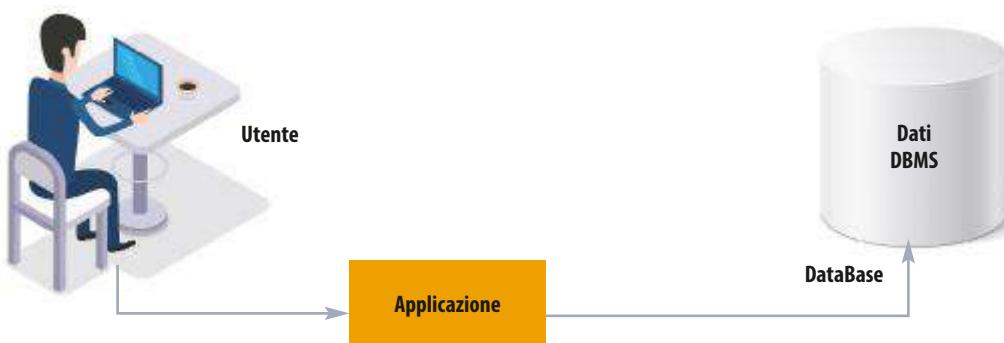
La principale caratteristica del database è quella di considerare solo i dati, cioè di separare completamente i dati dalle applicazioni.

Nei programmi che gestiscono dati scritti con i linguaggi procedurali abbiamo sempre definito per prima cosa il tipo di dato strutturato che rappresenta il record e questo nuovo tipo di dato è stato successivamente utilizzato per memorizzare le informazioni nell’archivio ed elaborarle estraendone informazioni. Senza rendercene conto, abbiamo separato la “manipolazione” dei dati dalla loro definizione.

Un passo fondamentale nella definizione dei dati di un database consiste nella descrizione di come sono formati i record, ovvero il numero, il nome e i tipi di dato che essi contengono, oltre ad altre informazioni complementari.

Dato che ogni applicazione deve utilizzare gli stessi archivi e, come detto prima, deve essere possibile modificare il formato dei record senza modificare le applicazioni, nasce la necessità di “includere” la descrizione dei dati con i dati stessi, in modo che qualunque programma li possa interpretare correttamente.

ESEMPIO

Se si prova ad aprire un qualunque file di dati con un editor di testo si scopre che non è più possibile (o è difficoltoso) interpretare i dati che vi sono salvati. Ciò significa che la massa dei dati, presa in modo autonomo dalla loro definizione, diviene sostanzialmente inutile.

Nei database, la definizione dei dati e i dati stessi sono salvati all’interno dello stesso database.

Lo schema di progettazione per le applicazioni con un archivio convenzionale risulta molto diverso rispetto allo schema di progettazione di un database, in quanto nel secondo caso sia i dati sia la loro definizione sono salvati all’interno dello stesso contenitore (il database).
Se si usano archivi convenzionali, infatti, la definizione dei dati è parte integrante dell’applicazione.

Con l'utilizzo di un database la definizione della struttura dei dati è invece indipendente dall'applicazione.

Nel secondo caso è tuttavia necessario stabilire una interfaccia tra le diverse applicazioni e i dati che sia in grado di interpretare la definizione della loro struttura, di recuperarne la posizione fisica e di gestirli.
Questo componente si chiama DBMS.

Si definisce DBMS (*Database Management System*) il sistema di gestione del database visto nel suo complesso. Il DBMS si preoccupa di gestire interamente i dati, compresa la loro definizione e il modo in cui vengono fisicamente archiviati.

Esamineremo nel paragrafo successivo tutte le funzioni di un DBMS, ma proponiamo di seguito un esempio per meglio comprendere la necessità di separazione tra dati (e loro definizione) e applicazioni.

ESEMPIO

Un cliente necessita di un'applicazione per la gestione di un magazzino che consenta agli utenti di lavorare contemporaneamente su parti diverse della gestione (carico/scarico). Chi si occupa del carico (arrivo della merce) utilizza un PC da tavolo tradizionale. Chi invece si occupa dello scarico (vendita della merce) utilizza un terminale a radiofrequenza per leggere i codici a barre degli oggetti che vengono prelevati dal magazzino.
In prima analisi si potrebbe pensare di utilizzare un programma tradizionale che memorizza i dati in file di record.
Resta tuttavia il problema dei terminali a radiofrequenza: questi sono dei piccoli computer a tutti gli effetti che, però, non offrono la stessa flessibilità dei PC tradizionali e con ogni probabilità supporteranno un linguaggio di programmazione (per esempio, Java o C#) diverso da quello nel quale è stata scritta l'applicazione del PC fisso.

Da questo esempio si può evincere come il tenere ben distinti l'applicazione e i dati (con la loro definizione) diventa non solo importante, ma indispensabile: se così non fosse, fornire l'applicazione al nostro cliente diventerebbe o impossibile o perlomeno molto più oneroso.

Vediamo adesso un secondo esempio, che ci fa capire ancora meglio come sia vantaggioso utilizzare un database rispetto a un archivio organizzato in modo tradizionale.

ESEMPIO

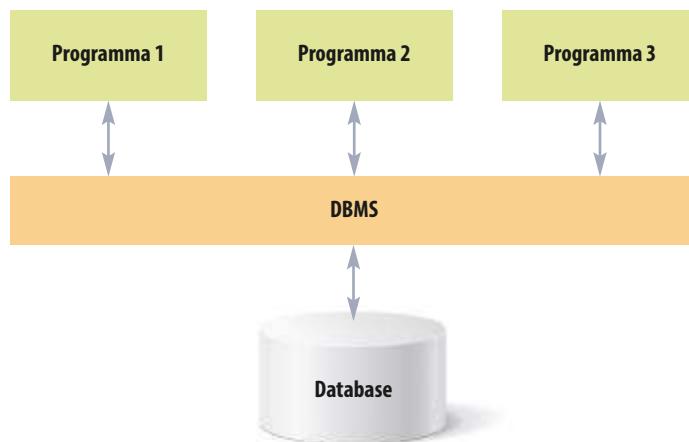
Supponiamo di voler registrare le informazioni relative agli ordini di materiale elettrico fatti da alcuni clienti di un'impresa. In base alle più tradizionali tecniche di gestione degli archivi è possibile strutturare i dati in due archivi, Clienti e Ordini, contenenti rispettivamente le informazioni relative ai clienti e quelle relative agli ordini fatti dai clienti.

CLIENTI

Cod_Cliente	Cognome	Nome	Indirizzo	Città
010	Rossi	Pino	Via Milano, 15	Como
020	Verdi	Giuseppe	Via Italia, 1	Milano
030	Neri	Piero	Via Colombo, 3	Varese

ORDINI

Cod_Cliente	Cod_Articolo	Descrizione	Prezzo	Quantità
010	M03	Lampadina 1000W	2,00	3
010	M12	Deviatore	3,00	1
010	M04	Antenna	25,00	3
020	M06	Trasformatore	60,00	2
020	M03	Lampadina 1000W	2,00	2
020	M12	Deviatore	3,00	1
030	M03	Trasformatore	60,00	2

Possiamo subito individuare alcuni problemi legati al tipo di organizzazione dell'archivio:
- esistono dati ripetuti: la descrizione dell'articolo viene ripetuta per ogni movimento (ridondanza dei dati);
- nascono problemi di incongruenza dei dati a causa della ridondanza: se un dato di un articolo viene modificato, tale modifica dovrà essere apportata a tutti i record che contengono quell'articolo (si pensi, per esempio, al caso dell'aumento di prezzo della lampadina, oppure alla modifica della descrizione di un prodotto);
- nascono problemi di inconsistenza a causa dell'incongruenza: si potrebbero avere due valori diversi per lo stesso dato senza poter quindi risalire al valore corretto (se venisse aggiornato solo un prezzo della lampadina non si è in grado di sapere "effettivamente" quanto costa una lampadina!).

Questi problemi, dovuti al fatto che i dati contenuti negli archivi non sono organizzati in modo integrato tra loro, vanno ad aggiungersi a quelli prima descritti in merito alle applicazioni, cioè:
- impossibilità di modificare la struttura di un record senza generare di conseguenza a cascata un insieme di modifiche in tutti i programmi che utilizzano quel file (dipendenza logica);
- scarsa flessibilità in caso di nuove esigenze che potrebbero essere irrealizzabili a causa della struttura degli archivi: l'organizzazione scelta per memorizzare i dati vincola infatti il programmatore nell'uso delle operazioni che si possono effettuare sugli archivi (dipendenza fisica);
- i dati contenuti negli archivi possono essere trattati solo elaborando i file record per record.

La teoria dei database introduce una nuova metodologia di organizzazione degli archivi di dati con l'obiettivo di superare i limiti appena descritti.

Funzioni di un DBMS

Come abbiamo detto nel paragrafo precedente, il DBMS (*Database Management System*) si occupa della gestione del database gestendo interamente i dati, compresa la loro definizione e il modo in cui vengono fisicamente archiviati: si colloca cioè tra i programmi applicativi e i file e gestisce i dati inserendoli fisicamente sui supporti magnetici e recuperandoli da essi man mano che vengono richiesti dalle applicazioni che operano sulla base di dati.

Le caratteristiche fondamentali di un DBMS possono essere così sintetizzate:
– deve gestire grandi quantità di dati; i dati hanno dimensioni maggiori della memoria centrale e i DBMS devono gestire i dati in memoria secondaria;
– deve garantire la condivisione dei dati che devono poter essere usati da applicazioni e utenti diversi secondo proprie modalità;
– deve garantire la persistenza dei dati che devono durare nel tempo, oltre le singole applicazioni.

Inoltre un DBMS deve offrire all'utente diverse funzionalità che lo aiutino nella gestione e lo sviluppo di applicazioni che fanno uso del database (per esempio interfacce grafiche per l'amministrazione).
Tralasciando questo ultimo aspetto, che dipende dal produttore del DBMS, approfondiamo i punti precedenti.

Gestione

Gestire una grande quantità di dati non significa solo riuscire a manipolare grandi spazi su disco o su memoria, quanto piuttosto prestare particolare attenzione ai problemi di efficienza. Quando in un insieme di archivi sono disponibili milioni di dati, infatti, la loro utilità viene meno se ogni volta che bisogna accedere a un particolare dato è necessario attendere un tempo indefinito. I ritmi lavorativi odierni ci impongono di sviluppare applicazioni che abbiano bisogno solo di pochi secondi per fornire una soluzione o un'elaborazione. Il DBMS, pertanto, non deve assolutamente rappresentare un collo di bottiglia nei tempi di elaborazione. Allo stesso tempo, tuttavia, il DBMS deve permettere a più utenti di consultare i dati. Anche se questo requisito potrebbe sembrare piuttosto banale, a una più attenta riflessione si capisce quanto complicato sia il lavoro da esso svolto.

Condivisione

Gestire grandi quantità di dati, garantendo velocità di elaborazione e contemporaneamente permettere a più utenti di condividere le stesse informazioni, non solo sono attività che possono diventare incompatibili tra di loro, ma introducono anche la necessità di coordinare gli accessi per evitare di fornire a un utente dati errati o non aggiornati.

Persistenza

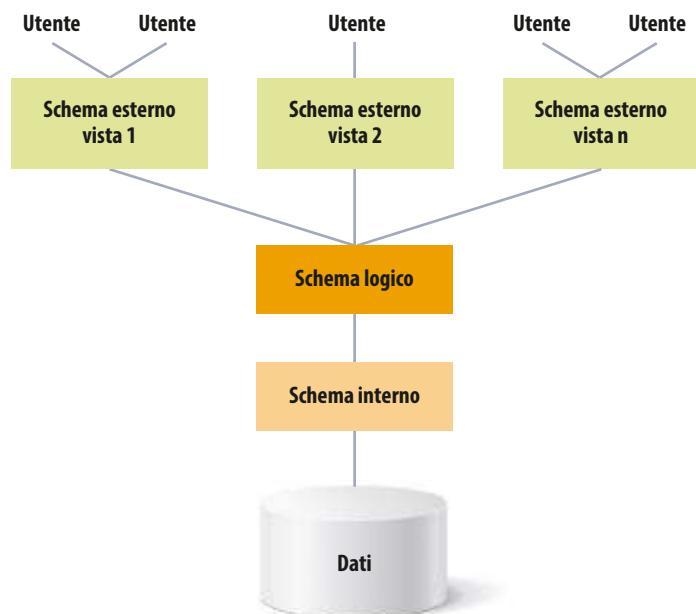
La persistenza dei dati, inoltre, richiede che il DBMS compia ancora un'altra operazione, ovvero fornisca quei meccanismi che permettono di assicurare l'affidabilità dei dati. La presenza di più utenti comporta infatti anche più possibilità che male intenzionati o incauti accedano a dati su cui non hanno privilegi di accesso. Il DBMS deve allora anche gestire il controllo degli accessi per assicurare che i dati siano visibili solo da particolari utenti o gruppi di utenti.

Senza entrare in dettagliate spiegazioni teoriche circa il funzionamento un DBMS, per comprendere come funziona ci limiteremo a usarlo.

PER SAPERNE DI PIÙ

DBMS COMMERCIALI E OPEN SOURCE
Di seguito elenchiamo alcuni tra i DBMS attualmente più diffusi, ripartibili in commerciali e open source.

Commerciali		Open source
– Oracle	– Sybase	– MySQL
– MS SQL Server	– Informix	– PostgreSQL
– MS Access	– Ingres	
– IBM DB2		

Architettura standard a tre livelli per DBMS (ANSI/SPARC)

ANSI/SPARC
ANSI/SPARC è l'acronimo di *American National Standards Institute, Standards Planning And Requirements Committee*: nel 1975 fu proposto un modello astratto come schema di architettura standard per il database, al quale fu dato questo nome e che ancora oggi è un modello di riferimento.

L'architettura standard del DBMS (**ANSI/SPARC**) si articola su tre livelli, che descriviamo qui di seguito.

- Schema esterno: descrizione di una porzione della base di dati di interesse in un modello logico ("viste" parziali, derivate, anche in modelli diversi).
- Schema logico: descrizione dell'intera base di dati nel modello logico adottato dal DBMS.
- Schema interno (o fisico): rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione.

L'utente non accede allo schema logico nella sua interezza ma solamente a quella porzione che interessa alla sua applicazione, dato che "al di sopra dello stesso database" vengono mandate in esecuzione diverse applicazioni: anche all'interno della singola applicazione viene utilizzata sempre solo una "parte" del database, che prende il nome di vista.

ESEMPIO

In questo esempio possiamo osservare come le prime due tabelle contengono i dati rispettivamente di materie scolastiche e delle aule dell'istituto scolastico.

AULE		
Nome	**Edificio**	**Piano**
NB23	Centrale	Terra
AU12	Ala_est	Primo
XY12	Ala_est	Secondo
ZZ12	Ala_sud	Terzo

MATERIE		
Corso	**Docente**	**Aula**
Informatica	Verdi	AU12
Educazione fisica	Rossi	XY12
Matematica	Bianchi	NB23
Lingua inglese	Gialli	XY12

La terza tabella è "ricavata" dalle altre due: non è una tabella che fisicamente è memorizzata nel database, ma una particolare vista dei dati in esso presenti.

MATERIE/AULE			
Corso	**Aula**	**Edificio**	**Piano**
Informatica	AU12	Ala_est	Primo
Educazione fisica	XY12	Ala_est	Secondo
Matematica	NB23	Centrale	Terra
Lingua inglese	XY12	Ala_est	Secondo

Vediamo adesso un secondo esempio.

ESEMPIO

In questo caso nella prima tabella sono riportati i dati anagrafici di alcune ditte, mentre nella seconda tabella sono presenti i dati relativi alle città.

RESIDENZA		
Città	**Sigla**	**Cap**
Brescia	BS	25100
Milano	MI	20100
Bergamo	BG	24100
Como	CO	22100

CLIENTI		
Ragione Sociale	**Indirizzo**	**Città**
Dadi e viti S.p.A.	Via Piave, 15	Bergamo
Tutto per il bricolage	V.le Arno, 155	Milano
Fai da te S.p.A.	Via Verdi, 12	Como
La casa del cacciavite	Via Rossi, 8	Brescia

La terza tabella, che risulta essere più completa, viene ricavata della due precedenti "unendo i dati" in modo da poterla utilizzare per esempio per inviare la corrispondenza.

Ragione Sociale	Indirizzo	Città	Cap
Dadi e viti S.p.A.	Via Piave, 15	Bergamo	24100
Tutto per il bricolage	V.le Arno, 155	Milano	20100
Fai da te S.p.A.	Via Verdi, 12	Como	22100
La casa del cacciavite	Via Rossi, 8	Brescia	25100

I DBMS che verranno descritti in seguito rispettano l'architettura ANSI/SPARC: nella nostra trattazione esamineremo le tecniche di progettazione dei database per arrivare alla definizione dello schema logico e, quindi, alla strutturazione delle tabelle dello schema interno con la descrizione delle schema relazionale.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 In un sistema informatico quali tra i seguenti sono componenti software?
- **a** Archivi
- **b** Strumentazione
- **c** Applicazioni
- **d** Supporti fisici

2 Che cosa significa l'acronimo EDP?
- **a** Electronic Data Program
- **b** Electronic Disk Processing
- **c** Electronic Data Processing
- **d** Electronic Disk Program

3 Quale delle seguenti affermazioni sui database è vera?
- **a** Un database è una specie di memoria digitale intelligente
- **b** Un database permette di migliorare la compressione dei dati
- **c** Nei database la definizione dei dati e i dati sono salvati all'interno dello stesso database
- **d** In un database è possibile integrare i programmi con i dati

4 Per quali dei seguenti scopi può essere utilizzato un database?
- **a** Per la scrittura di un documento lungo e strutturato come un libro
- **b** Per gestire i propri contatti
- **c** Per la gestione di una biblioteca
- **d** Per migliorare le prestazioni del computer

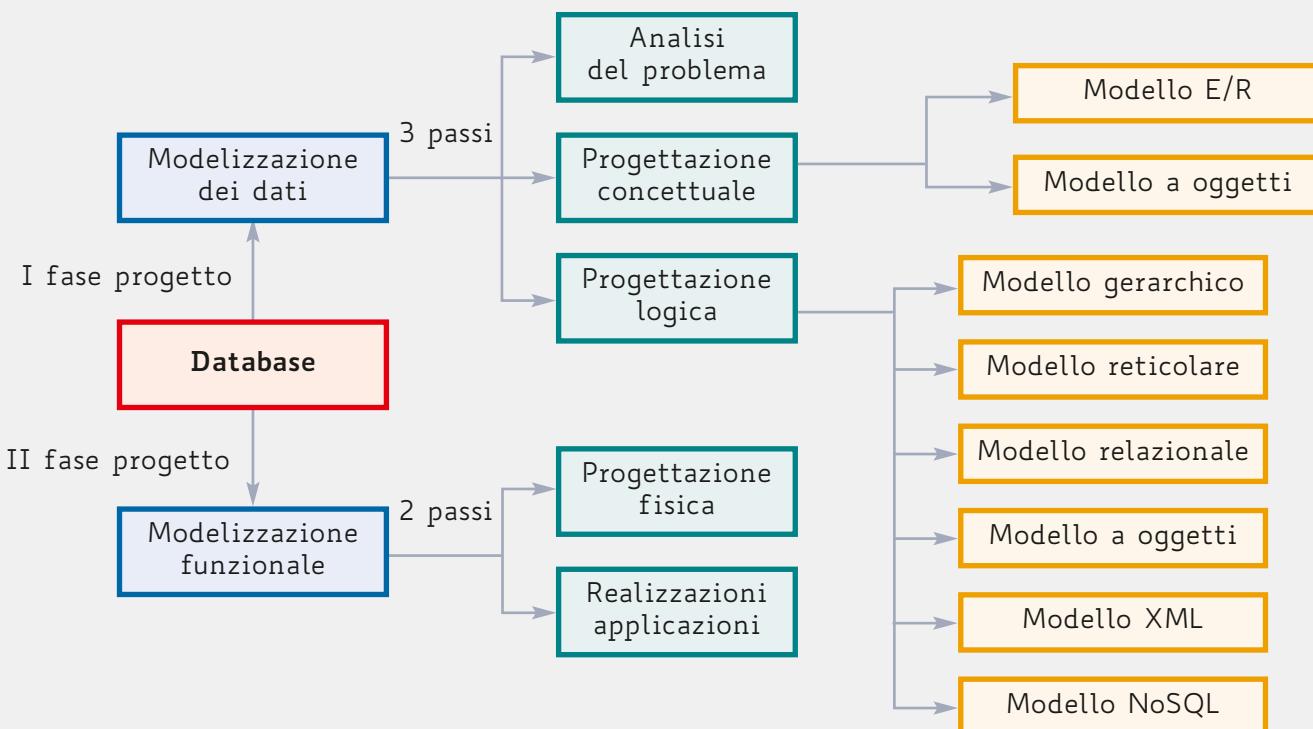
5 Qual è il più importante vantaggio dei DB?
- **a** L'applicazione e i suoi dati sono indipendenti
- **b** I dati occupano meno spazio su disco
- **c** I DBMS gestiscono più velocemente i dati
- **d** Facilitano la duplicazione dei file

6 Quale tra le seguenti non è una funzione del database?
- **a** Deve gestire grandi quantità su dati
- **b** Deve garantire la correttezza dei dati
- **c** Deve garantire la condivisione
- **d** Deve garantire la persistenza

VERO/FALSO

1 In un database l'applicazione e i suoi dati sono indipendenti l'uno dagli altri. V F
2 I database risolvono i problemi di inconsistenza causati dell'incongruenza dei dati. V F
3 I database risolvono i problemi di incongruenza dei dati a causa della ridondanza. V F
4 I database risolvono i problemi di inconsistenza a causa della ridondanza. V F
5 I database hanno minor indipendenza logica rispetto agli archivi tradizionali. V F
6 I DBMS sono componenti hardware. V F
7 Oracle è un database open source. V F
8 L'architettura standard per il DBMS è a quattro livelli. V F
9 Lo schema logico descrive l'intera base di dati nel modello logico adottato dal DBMS. V F
10 L'indipendenza dei dati è indispensabile solo a livello logico. V F

RISPOSTA APERTA

1 Che cosa si intende per sistema informativo e informatico?
2 Quali sono i limiti degli archivi tradizionali?
3 Cosa si intende per DBMS?
4 Quali sono le funzioni di un DBMS?
5 Come è organizzata l'architettura ANSI/SPARC?

Progettazione concettuale e logica

IN QUESTA LEZIONE IMPAREREMO...

- quali sono le fasi della progettazione di un database
- che cos'è lo schema concettuale dei dati
- in che cosa consiste il modello logico dei dati
- quali sono i diversi modelli logici esistenti

MAPPA CONCETTUALE

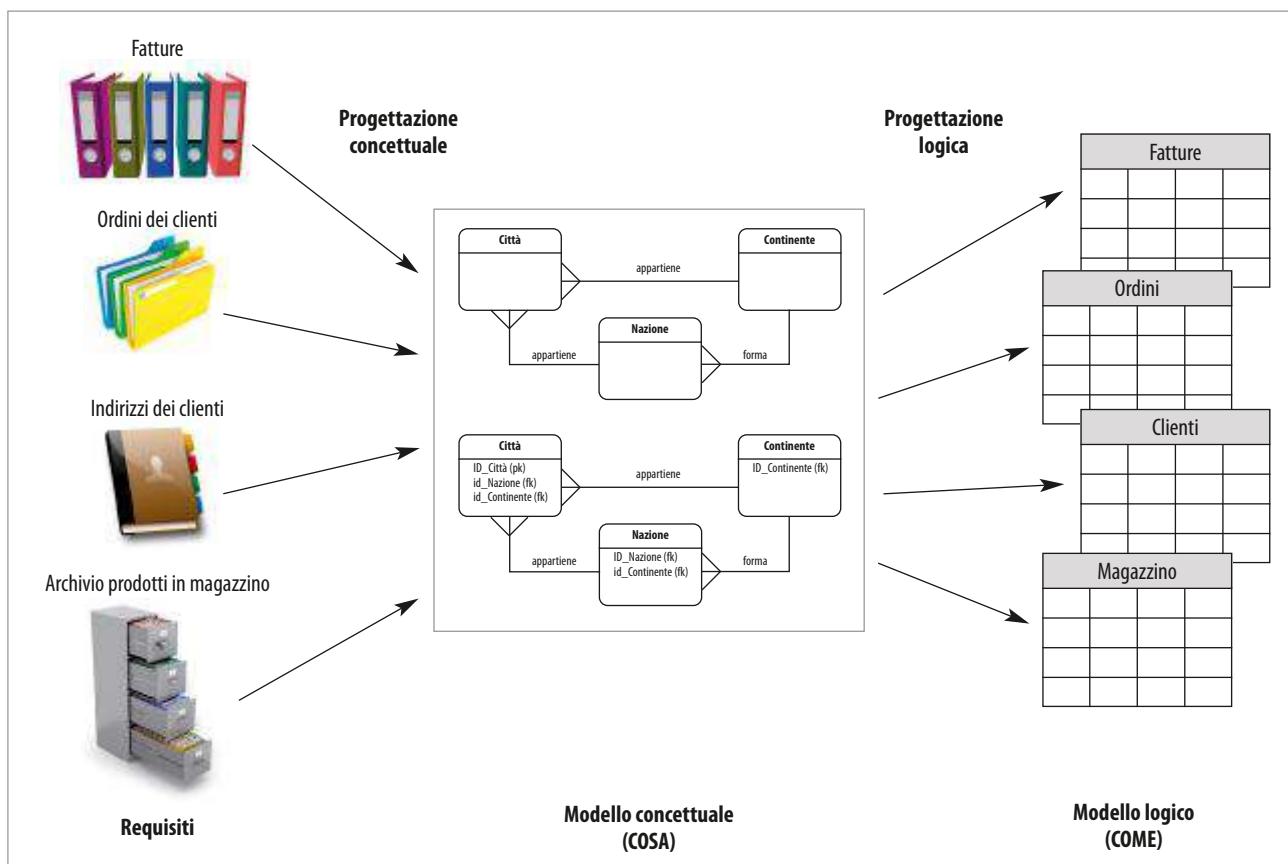
didattica inclusiva

Database
I fase progetto → Modellizzazione dei dati
3 passi: Analisi del problema; Progettazione concettuale; Progettazione logica
Progettazione concettuale → Modello E/R; Modello a oggetti
Progettazione logica → Modello gerarchico; Modello reticolare; Modello relazionale; Modello a oggetti; Modello XML; Modello NoSQL
II fase progetto → Modellizzazione funzionale
2 passi: Progettazione fisica; Realizzazioni applicazioni

Generalità

La progettazione di un database segue fasi e regole ben precise.

Progettare un database significa progettarne le strutture, prima logiche e quindi fisiche, in modo che possano accogliere nel modo migliore possibile i dati di cui un utente ha bisogno.

I passi principali per progettare un database si possono così schematizzare:
1 analisi del problema;
2 progettazione concettuale del database (modello E-R);
3 progettazione logica del database (schema logico);
4 progettazione fisica e implementazione;
5 realizzazione delle applicazioni.

Ognuno di questi passi presenta delle criticità e implica un insieme di operazioni anche complesse: per esempio, ciò che si cela dietro il semplice passo “implementazione” racchiude tutta la fase di sviluppo del database, dell’applicazione, i test di funzionamento di entrambi, il collaudo ecc.

I cinque passi sopracitati si possono poi ulteriormente accorpare in:
- modellazione dei dati vera e propria, che include l’analisi e la progettazione concettuale e logica del database;
- modellazione funzionale, che include la progettazione fisica e la realizzazione delle applicazioni.

Il modello concettuale descrive cosa deve essere rappresentato, mentre il modello logico descrive come sono organizzati i dati.

Fatture
Ordini dei clienti
Indirizzi dei clienti
Archivio prodotti in magazzino
Requisiti

Progettazione concettuale

Città — appartiene — Continente
Nazione — appartiene — forma

Città: ID_Città (pk), id_Nazione (fk), id_Continente (fk)
Continente: ID_Continente (fk)
Nazione: ID_Nazione (fk), id_Continente (fk)
appartiene / appartiene / forma

Modello concettuale (COSA)

Progettazione logica

Fatture
Ordini
Clienti
Magazzino

Modello logico (COME)

Detto altrimenti, la modellazione dei dati consiste nella progettazione delle tabelle del database, mentre la modellazione funzionale consiste nell’implementazione delle tabelle e nella creazione delle funzioni che accedono ai dati.

Modellazione dei dati

In informatica l’astrazione è un aspetto fondamentale perché ci permette di creare dei modelli su cui in seguito vengono costruite le applicazioni. La fase di analisi di ogni progetto, indipendentemente dalla sua complessità, serve proprio per modellare il problema e quindi adottare una soluzione tecnica: questa stessa metodologia viene applicata anche ai dati del database.

Un modello di dati consiste in una rappresentazione astratta delle strutture dei dati di un database. L’atto di creazione di un modello prende il nome di modellazione dei dati (in inglese, *data modeling*).

La modellazione dei dati serve per tradurre i dati dal punto di vista dell'utente al punto di vista dell'applicazione/database, cioè per trasporli dal mondo reale al mondo informatico.

Analisi e progettazione concettuale

L'analisi preliminare per la modellazione dei dati avviene solitamente cercando di individuare le esigenze del cliente o il dominio dell'applicazione, cioè quali informazioni devono essere salvate e in che modo queste informazioni verranno manipolate dall'utente.

Bottom-up e top-down

Le tecniche top-down e bottom-up sono metodologie informatiche per la risoluzione dei problemi e la progettazione del software a partire da situazioni problematiche e/o complesse.
Nel modello top-down si formula inizialmente una visione generale del sistema, ovvero se ne descrive la finalità principale senza scendere nel dettaglio delle sue parti, quindi, con affinamenti successivi, si arriva a dettagliare ogni singola operazione elementare.
Nella progettazione bottom-up si invece inizia la progettazione dalle parti individuali del sistema, specificandole nel dettaglio, per poi connetterle tra loro in modo da formare componenti più grandi fino a realizzare un sistema completo.

Le tecniche di analisi del problema sono da considerarsi valide e applicabili anche alla progettazione dei database: naturalmente bisognerà prestare maggiore attenzione al dato e quindi spesso la tecnica **bottom-up** è da preferirsi a quella **top-down**.

Al termine della analisi inizia la prima fase di modellazione, che è quella concettuale; per attuarla, si può far ricorso ai due seguenti modelli:
- modello Entità-Relazione;
- modello a Oggetti.

Benché il secondo modello stia cominciando a rivelarsi interessante da un punto di vista commerciale e non solamente accademico, la stragrande maggioranza delle applicazioni esistenti ricorre all'approccio Entità-Relazione (E-R), che è pertanto quello che utilizzeremo nei nostri progetti descrivendolo nel dettaglio a partire dalla prossima lezione.

Il frutto della modellazione dei dati consiste nel diagramma Entità-Relazione (diagramma E-R), che rappresenta in modo grafico e facilmente leggibile le strutture dei dati.

Può capitare di trovarsi a lavorare a progetti di una certa portata che coinvolgono programmatori o analisti di altri Paesi: la comunicazione tra colleghi può in questi casi risultare difficile a causa delle barriere linguistiche. Il modello Entità-Relazione consente di superare questo ostacolo in quanto è assolutamente indipendente dal linguaggio scritto o parlato e permette quindi a tutti di comprendere la struttura del database.

Di norma, al modello Entità-Relazione viene affiancato un documento tecnico che descrive in maggior dettaglio, usando un linguaggio naturale, i concetti espressi graficamente dal modello Entità-Relazione.

La fase di modellazione dei dati, che è di primaria importanza, è però anche una delle fasi più difficili, laboriose e "noiose"nella creazione di un'applicazione che si interfaccia a un database:viene pertanto "naturale" tentare di ridurla al minimo, per iniziare "al più presto" a programmare.
A "parziale giustificazione" del fatto che si tenda a ridurre i tempi di modellazione vi è la considerazione che,mentre in un mondo ideale si può liberamente disporre del tempo per fare qualsiasi cosa e produrre software di qualità, nel mondo reale i tempi destinati allo sviluppo sono generalmente molto stretti e, poiché la fase di modellazione richiede al contrario una lunga gestazione, si cerca di "comprimerla" al massimo.

Tempi di sviluppo e costi creano solitamente qualche attrito tra i reparti tecnici e i reparti commerciali delle aziende di sviluppo software.

Se anche voi vi siete chiesti se non sia possibile saltare a piè pari la modellazione concettuale dei dati, allora provate a rispondere a questa domanda: "Vi fidereste a passare su un ponte sapendo che chi l'ha costruito non disponeva dei piani di costruzione?"

Come abbiamo appena visto, il primo scopo del modello Entità-Relazione è quello di fornire la rappresentazione grafica di tutti gli oggetti che fanno parte di un database, cosi che il flusso delle informazioni possa essere seguito e verificato prima di sviluppare l'applicazione.

In secondo luogo, questo modello può essere usato dagli sviluppatori per creare il database fisico e tutti gli oggetti che ne fanno parte.

Una cattiva modellazione ha impatti che possono essere devastanti dal punto di vista dell'applicazione che usa i dati, fino a rendere il flusso dei dati completamente inutilizzabile.
Anche se la modellazionerichiede tempo, se viene eseguita in modo corretto permette di sviluppare l'applicazione più velocemente e in modo più flessibile, così da agevolare eventuali modifiche successive del flusso delle informazioni.

Riassumendo, un buon progetto concettuale, per definirsi tale, deve essere caratterizzato da:
- correttezza: uso corretto degli strumenti;
- completezza: tutti gli aspetti rilevanti della realtà devono essere modellati;
- chiarezza: il modello deve essere leggibile e rappresentare le informazioni in maniera comprensibile;
- indipendenza dallo strumento informatico che verrà utilizzato.

Modellazione logica

Una volta approntato il modello concettuale, si procede alla definizione del modello logico dei dati, che consiste in uno schema realizzato in funzione delle caratteristiche del sistema di gestione del database che si intende utilizzare (la determinazione delle strutture dei dati, il linguaggio per sviluppare le applicazioni).

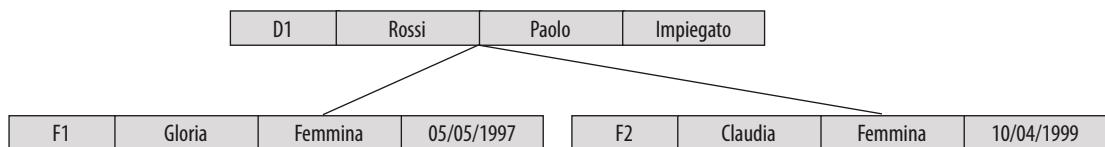
Il modello logico è "più vicino" alla rappresentazione informatica dei dati: per ottenerlo, è necessario tradurre lo schema concettuale attraverso l'applicazione di un insieme di regole ben definite.

Il modello logico deve essere:
- indipendente dalle strutture fisiche;
- utilizzato dai programmi applicativi.

Nel tempo, si sono succeduti diversi tipi di modelli logici:
- gerarchico, rappresentabile tramite un albero (anni '60);
- reticolare, rappresentabile tramite un grafo (anni '60);
- relazionale, attualmente il più diffuso, rappresentabile mediante tabelle e relazioni tra esse (anni '70);
- a oggetti, che è una estensione alle basi di dati del paradigma "Object-Oriented", tipico della programmazione a oggetti (anni '80);
- XML, molto utilizzato come strumento per l'esportazione di dati tra diverse applicazioni (anni '90).

Esaminiamoli più nel dettaglio nei paragrafi che seguono.

Modello gerarchico

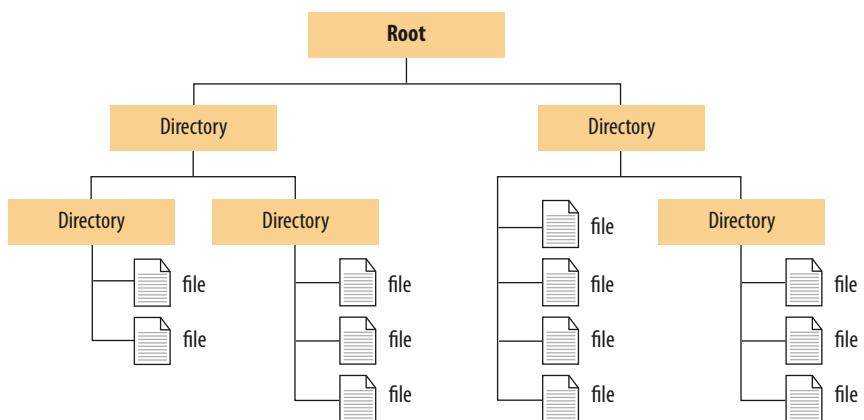
Il primo modello ad affermarsi sul mercato è stato il modello di database gerarchico.
Nel modello gerarchico i dati sono organizzati secondo strutture ad albero che rappresentano la gerarchia degli elementi presenti nell'archivio.
La radice è il record principale del database, da cui partono uno o più sottoalberi a esso simili: naturalmente il numero dei figli è variabile.
Ogni elemento prende il nome di segmento e il modello gerarchico permette di rappresentare i dati sfruttando la relazione tra segmenti padre e segmenti figli, realizzando quella che prende nome di relazione 1 a n o relazione uno-a-molti, si indica sinteticamente 1:n oppure (1, n).

Nella relazione uno-a-molti ogni padre può avere molti figli, ma ogni figlio può avere un solo padre.

Vediamo un esempio di modello gerarchico.

ESEMPIO

Supponiamo di avere un archivio con i dati dei dipendenti di un'azienda, dove nella tabella principale abbiamo come campi nome, cognome, qualifica. Oltre ai dati dei dipendenti, nell'archivio troviamo memorizzati anche i dati dei figli dei dipendenti (per esempio, i campi nome, sesso, data di nascita).
La tabella dipendenti è composta dal segmento padre e la tabella figli dal segmento figlio.
Naturalmente un dipendente può avere più figli, quindi si crea la gerarchia dipendenti-figli.

Nel database ogni dipendente avrà un suo albero.

Vediamo adesso un altro esempio, dove possiamo individuare una "organizzazione" di elementi sempre secondo il modello gerarchico.

ESEMPIO

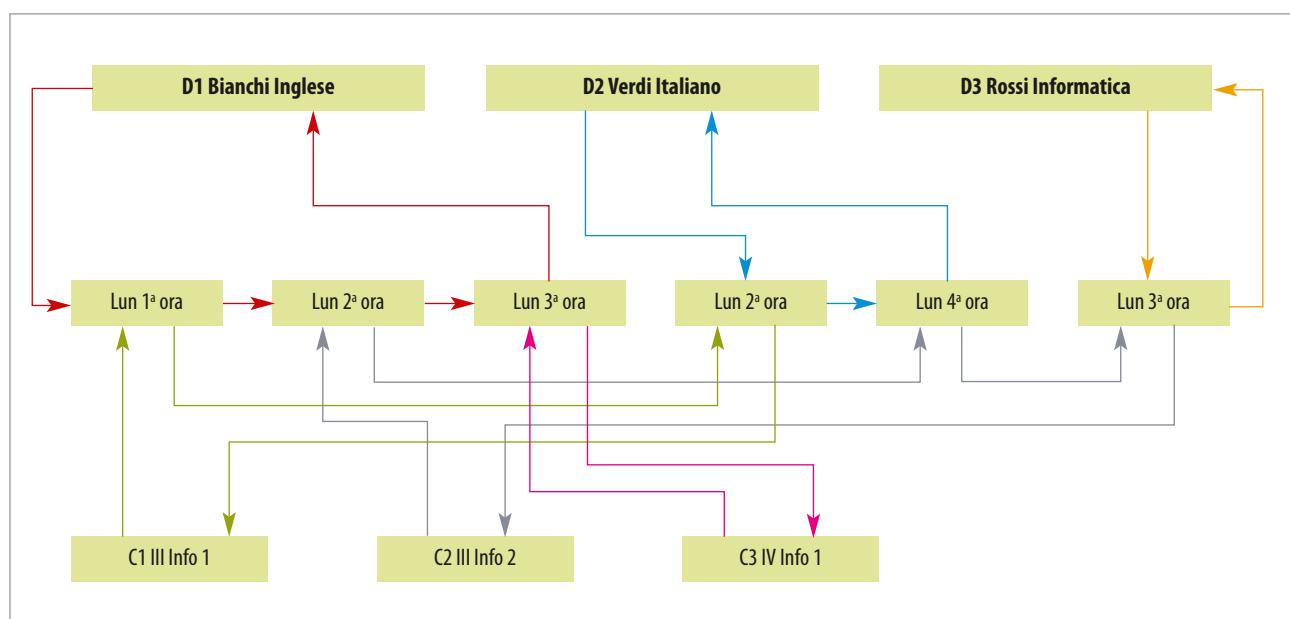
Un modello gerarchico che utilizziamo quotidianamente è rappresentato dall'organizzazione dei file del file system del nostro PC, organizzato in directory e sottodirectory. Questo modello era usato nei primi DBMS per mainframe; in seguito i DBMS reticolari hanno sostituito quelli gerarchici per poi essere sostituiti da quelli relazionali (anche chiamati **RDBMS**).

RDBMS
Il Relational Database Management System (RDBMS), vale a dire il sistema per la gestione di basi di dati relazionali, è stato introdotto da Codd e indica un DBMS basato sul modello relazionale.

In un database organizzato secondo il modello gerarchico è semplice recuperare le informazioni quando i dati sono proprio di natura gerarchica, come negli esempi proposti, mentre diviene molto complesso estrarre i dati secondo altri criteri (per esempio, per estrarre tutte le figlie femmine dei dipendenti è necessario visitare tutti gli alberi).

I principali svantaggi del modello gerarchico possono essere così sintetizzati:
- tra lo schema logico e la realizzazione fisica esiste una dipendenza stretta e vincolante;
- le operazioni di ricerca non sono efficienti in quanto sono visite ad alberi generici e solo nel caso siano di tipo gerarchico sarebbero di semplice realizzazione.

Modello reticolare

Alla base del modello reticolare ci sono le strutture dati a grafo, dove mediante puntatori è possibile accedere ai dati più facilmente, senza i vincoli rigidi della struttura gerarchica. Possiamo quindi considerare il modello reticolare come un'estensione del modello gerarchico, dove non esiste alcuna radice, ma ogni nodo può essere il punto di partenza per raggiungere un determinato campo.

Nel modello reticolare ogni elemento è costituito da un record che può connettersi con altri n record: è quindi possibile stabilire delle relazioni multiple del tipo n con n indicate con n:n oppure (n, n), impossibili nel modello gerarchico.

Per poter realizzare le connessioni tra i diversi record vengono utilizzati particolari record che prendono il nome di record connettori.

ESEMPIO

Supponiamo di voler rappresentare tramite database i Docenti e le Classi nelle quali insegnano. Il modello reticolare che ne risulta è il seguente.

Una volta che il database è stato realizzato risulta tuttavia molto complesso apportare delle modifiche, in quanto si rischia di dover riscrivere tutte le applicazioni che già lo utilizzano.

I principali svantaggi del modello reticolare sono i seguenti:
- i link sono realizzati con i puntatori, quindi vi è uno spreco di spazio per le memorie esterne;
- per realizzare due reticoli indipendenti è necessario duplicare i dati introducendo un'inutile ridondanza;
- se i dati non sono tra loro direttamente connessi la loro ricerca risulta difficoltosa;
- è estremamente rigido in caso di modifiche successive alla sua creazione.

Modello relazionale

Edgar F. Codd
Edgar Frank "Ted" Codd è stato il fondatore della teoria delle basi di dati relazionali: lavorò negli anni Sessanta-Settanta per l'IBM e pubblicò nel 1970 il primo articolo nel quale presentò i database relazionali come "modello per l'archiviazione di grandi banche di dati".

Il modello relazionale è stato definito da **Edgar F. Codd** all'inizio degli anni '70 con l'obiettivo di non duplicare inutilmente le informazioni: consiste in un insieme di tabelle che possono essere connesse tra loro mediante, appunto, relazioni.

Alla base del modello relazionale c'è il concetto matematico di relazione tra insiemi.

Il modello relazionale usa come struttura dati fondamentale la relazione (o tabella) e per operare sul database definisce un numero ristretto di operazioni fondamentali.

Ogni tabella è composta da righe e colonne:
- le colonne sono i diversi campi (o proprietà);
- ogni riga corrisponde a un record.

I principali vantaggi del modello relazionale sono:
- l'indipendenza dei dati;
- il fatto che la rappresentazione logica dei dati non fa alcun riferimento a quella fisica;
- la collocazione dell'informazione nei campi e non in strutture fisiche come i puntatori.

Con indipendenza dei dati, come vedremo in seguito, si intende:
- indipendenza dalla struttura fisica. Eventuali modifiche apportate alla rappresentazione fisica dei dati (per esempio uso di una struttura di accesso piuttosto di un'altra) non comportano modifiche ai programmi applicativi esistenti;
- indipendenza dalla struttura logica. Eventuali modifiche apportate alla rappresentazione logica dei dati non devono comportare modifiche alle applicazioni esistenti che operano sul database, cioè deve essere possibile apportare modifiche alla struttura della base di dati senza modificare il software applicativo.

Il modello relazionale è attualmente il modello logico di dati più utilizzato e sarà descritto e studiato nelle prossime lezioni.

Tra i DB relazionali ricordiamo MySQL, SAPDB, Firebird, Access, Oracle.

Modello a oggetti (Object-Oriented)

I database a oggetti sono la nuova frontiera nella ricerca sui database per le caratteristiche di estendibilità proprie della programmazione *object-oriented*: hanno la possibilità di definire nuovi tipi di dati e comportamenti che vengono inglobati nell'oggetto stesso (classe); trovano utilizzo soprattutto nelle applicazioni multimediali che inglobano grande quantità di dati non numerici, come, per esempio, immagini, suoni o filmati.

I DBMS a oggetti prendono il nome di ODBMS (*Object Database Management System*) oppure OODBMS (*Object-Oriented Database Management System*): la tendenza odierna è quella di aggiungere alle caratteristiche dei sistemi a oggetti i vantaggi del modello relazionale, generando sistemi ibridi che uniscano i pregi di entrambi i modelli (ORDBMS, *Object-Relational Database Management System*).

I database object-oriented si fondano sugli stessi principi della programmazione a oggetti e quindi memorizzano nel database non solo i dati ma anche le operazioni che possono essere eseguite sugli stessi (metodi).

Tra i primi ODBMS ricordiamo Jasmine, sviluppato dalla Fujitsu insieme alla Computer Associates verso la fine degli anni '90 e integrato in VisualObject, uno dei primi linguaggi visuali che offriva la possibilità di creare delle classi per interfacciare in modo nativo il database Jasmine.

Oggi il più famoso tra gli ORDBMS è PostgreSQL: è completamente open source ed è probabilmente il più robusto del mondo Linux/Unix: è stato progettato e realizzato all'università della California a Berkeley sotto il nome di Post gres, ma venne poi ripreso da un gruppo esterno di sviluppatori che gli diedero il nome attuale, gli aggiunsero una personalizzazione del linguaggio SQL e lo rilasciarono open source.

Modello XML (eXtensible Markup Language)

L'XML non è propriamente un modello di database, ma va ricordato perché è divenuto uno strumento fondamentale per effettuare lo scambio delle informazioni tra DBMS diversi.
È un linguaggio simile all'HTML, con il quale condivide i markup (comunemente detti tag), ma

nel linguaggio XML i tag sono “liberi” e descrivono quanto racchiuso tra le parentesi <>, come definito dal programmatore.

L’XML può anche essere definito come un metalinguaggio che ha lo scopo di rappresentare contenuti testuali organizzati in modo gerarchico.

L’XML non è un linguaggio per il Web, così come non è un meccanismo per la sola rappresentazione di contenuti provenienti da basi di dati.
La principale caratteristica del linguaggio XML è che può rappresentare contenuti in un formato compatibile con qualsiasi sistema hardware e software in quanto si tratta di un semplice file in formato testo.

Il linguaggio XML è anche la modalità utilizzata dalla Pubblica Amministrazione per scambiare dati con i cittadini e le imprese, come nel caso della fatturazione elettronica, oppure del modello 730 precompilato.

NoSQL: una nuova proposta

Per far fronte alle nuove tendenze dell’informatica dominate negli ultimi anni dal social network e dal cloud computing, sono state proposte alcune alternative al modello relazionale: quelle che stanno riscuotendo i maggiori consensi appartengono alla famiglia NoSQL (“Not only SQL“). I DBMS progettati sotto questo “paradigma” non si oppongono a SQL, anzi, “prelevano” da SQL alcune funzionalità integrandole nel nuovo database che si propone di superare i suoi limiti in termini di elasticità e scalabilità.

SQL è l’acronimo di *Structured Query Language* ed è il linguaggio di programmazione usato per l’interrogazione e la gestione dei database relazionali (RDBMS): spesso i due termini – database relazionali e database SQL – vengono usati come sinonimi e da qui nasce il termine NoSQL.

NoSQL non individua comunque un linguaggio specifico o un modello di database, ma è il termine universalmente accettato per raggruppare un insieme di tecnologie per la persistenza dei dati che funzionano in modo sostanzialmente diverso dai database relazionali, non rispettando una o alcune delle caratteristiche proprie dei database relazionali, caratteristiche che possono essere così sintetizzate:
– utilizzo di tabelle e campi per memorizzare i dati;
– schema fisso delle tabelle: nome, elenco di campi, ognuno con i rispettivi tipi (stringhe di caratteri, numeri, date e dati binari) con una chiave primaria che identifica univocamente, senza ambiguità, una riga della tabella;
– presenza di una relazione tra due o più campi di tabelle, ossia di una condizione che lega tra loro le rispettive righe a cui i campi appartengono mediante chiavi esterne; la validità del legame è garantita dai vincoli di integrità referenziale;
– accesso ai dati (transazione) garantito con le proprietà ACID, cioè Atomicità, Consistenza, Isolamento e Durabilità.

I database NoSQL possono invece avere le caratteristiche più disparate: alcuni usano tabelle e campi ma senza schemi fissi, altri non utilizzano il modello relazionale con chiavi primarie ed

esterne, alcuni non gestiscono i vincoli di integrità referenziale e altri ancora non garantiscono transazioni ACID.

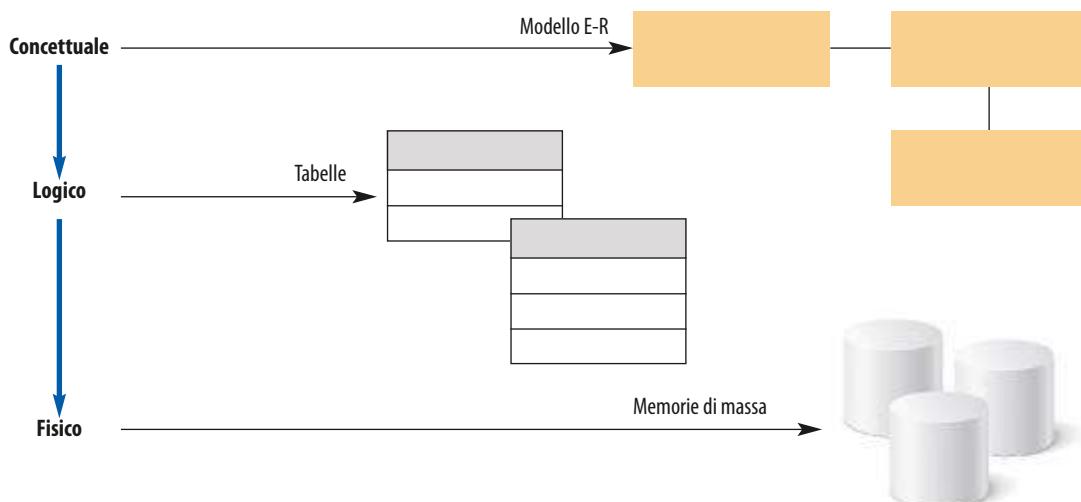
Questa tipologia di database si adatta a molti contesti, soprattutto quando si manipolano grandi quantità di dati eterogenei e senza uno schema, mentre non sono adatti nei casi in cui si devono gestire molte relazioni tra i dati e si vuole garantire l'integrità referenziale: in questi casi si utilizza il "buon vecchio" modello relazionale, come nella applicazioni ERP.

A titolo di esempio, ricordiamo i NoSQL che stanno avendo maggior successo (e diffusione).
- MongoDB è un database orientato ai documenti, cioè i dati vengono memorizzati in documenti che altro non sono se non oggetti complessi codificati in qualche modo e senza uno schema rigido.
- Redis, Memcached, HBase realizzano un database chiave-valore basato sul modello dell'array associativo e fanno corrispondere alle chiavi valori formati da famiglie di colonne anch'esse indicizzate.
- Neo4j implementa un database a grafo usando strutture con relazioni libere tra nodi del grafo.
- Cassandra, Big Table, SimpleDB utilizzano un database column-oriented che immagazzina dati in sezioni di colonne: la rapida aggregazione che permettono è stata apprezzata dai grandi produttori di motori di ricerca e social network, tanto che i alcune delle principali soluzioni di questo tipo sono state realizzate rispettivamente da colossi come Facebook, Google e Amazon.
- In Firebase non rimane nessuna traccia dell'approccio relazionale, cioè non vi sono tabelle né record: tutti i dati inseriti – che possono essere mappe, liste, stringhe, tipi numerici e booleani – vanno a costituire un **albero JSON**. Firebase è in grado di aggiornare i dati istantaneamente, sia se integrato in app web che mobile.

JSON

JSON, acronimo di *JavaScript Object Notation*, è un semplice formato per lo scambio di dati che è completamente indipendente dal linguaggio di programmazione. JSON è basato su due strutture, un insieme di coppie nome/valore e un elenco ordinato di valori, ed è particolarmente adatto all'interscambio di dati fra applicazioni client-server.

Implementazione e realizzazione

Al termine delle fasi di progettazione concettuale e logica, si passa all'ultima fase, in cui viene fisicamente realizzato il database (o sistema informativo) sul computer.
Questa fase prevede:
- il completamento, o modifica, dello schema logico in funzione dell'organizzazione fisica dei dati e dei meccanismi per operare su di essi (partizioni, puntatori, blocchi fisici, cluster, indici);
- il progetto e la realizzazione delle procedure atte a soddisfare le richieste specificate nel progetto utilizzando gli strumenti informatici di alto livello (linguaggi di programmazione e di interrogazione).

Le tre fasi della progettazione di un database possono essere così sintetizzate,visualizzando il "prodotto" di ciascuna di esse:
- modellazione concettuale: diagramma E-R;
- modellazione logica: tabelle e tracciati record;
- implementazione: definizione fisica delle tabelle.

La realizzazione del database comprende anche le operazioni di collaudo, che prevedono l'insieme dei test per verificare il corretto funzionamento del sistema.
Inoltre, durante il funzionamento del sistema, ricordiamo che costantemente vengono effettuate le operazioni di manutenzione, sia correttiva che adattativa.

Conclusioni

Indipendentemente dal modello che adotta, tre sono le funzionalità principali di un DBMS, che possono essere riassunte nella seguente definizione.

Un DBMS è un sistema software che gestisce efficientemente grandi quantità di dati, persistenti e condivisi.

Un DBMS deve quindi avere:
– la capacità di gestire dati non volatili, cioè in modo permanente;
– la capacità di accedere in modo efficiente a grandi quantità di dati.

La persistenza e la condivisione richiedono che un DBMS fornisca dei meccanismi per garantire l'affidabilità dei dati (*fault tolerance*), il controllo degli accessi e il controllo della concorrenza (accesso ai dati contemporaneo da parte di più utenti e/o applicazioni).

Deve inoltre garantire:
– la consistenza dei dati, che devono essere significativamente ed effettivamente utilizzabili nelle applicazioni aziendali;
– la sicurezza dei dati, che deve impedire che il database si danneggi;
– la segretezza e confidenzialità, affinché i dati possano essere consultati e/o modificati soltanto da chi sia debitamente autorizzato;
– l'integrità dei dati, cioè la loro conservazione senza perdite.

Le problematiche relative alla sicurezza sono state negli ultimi anni oggetto di grande attenzione, soprattutto in rapporto ai database connessi a Internet, dove è necessario proteggere i dati "appetibili" per gli hacker, in particolare quelli con carattere di estrema confidenzialità (numeri di carte di credito, dati finanziari, strategici ecc.), e impedire che i malintenzionati acquisiscano il completo controllo della macchina se non addirittura della intera infrastruttura di rete.

Tutti i più moderni DBMS adottano un'architettura di sicurezza basata su tre differenti livelli di protezione.
– Autenticazione: è la fase più delicata, dove l'utente che vuole accedere ai dati viene accreditato mediante la verifica dell'identità attraverso la richiesta di una password segreta e personale.
– Autorizzazione: è la fase seguente l'autenticazione, nella quale il sistema assegna i diritti all'utente per le risorse alle quali può avere accesso (lettura, scrittura ecc.).
– Auditing: in questa fase si adottano i mezzi idonei per garantire l'integrità delle informazioni e per identificare e riconoscere possibili abusi.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 La modellazione dei dati consiste:
- **a** nell'analisi del problema
- **b** nella progettazione delle tabelle del database
- **c** nell'implementazione delle tabelle
- **d** nella creazione delle funzioni che accedono ai dati

2 La modellazione funzionale consiste:
- **a** nell'analisi del problema
- **b** nell'implementazione delle tabelle
- **c** nella progettazione delle tabelle del database
- **d** nella creazione delle funzioni che accedono ai dati

3 Nel modello gerarchico le relazioni possono essere:
(2 risposte)
- **a** 1 a 1
- **b** 1 a molti (1, n)
- **c** molti a 1 (n, 1)
- **d** molti a molti (n, n)

4 L'acronimo ORDBMS significa:
- **a** Object-Relational Database Management Software
- **b** Object-Relative Database Management Software
- **c** Object-Relative Database Management System
- **d** Object-Relational Database Management System

5 La fase di autorizzazione all'accesso di un DB è la fase:
- **a** in cui si verifica l'identità attraverso la password segreta
- **b** seguente l'autenticazione
- **c** in cui si assegna la password all'utente
- **d** precedente all'autenticazione

6 Per i database NoSQL quale tra le seguenti affermazioni e falsa?
- **a** Neo4j implementa un database a grafo
- **b** Sono database che non ammettono SQL
- **c** MongoDB è un database NoSQ L orientato ai documenti
- **d** Sono utilizzati da Facebook, Google ed Amazon

ORDINAMENTO

1 Ordina nella corretta sequenza le operazioni da compiere per realizzare un database:
- **a** progettazione fisica e implementazione (_____)
- **b** progettazione concettuale del database (modello E-R) (_____)
- **c** analisi del problema (_____)
- **d** realizzazione delle applicazioni (_____)
- **e** progettazione logica del database (schema logico) (_____)

2 Ordina cronologicamente i seguenti modelli di DBMS:
- **a** XML (_____)
- **b** reticolare (_____)
- **c** gerarchico (_____)
- **d** a oggetti (_____)
- **e** elazionale (_____)

VERO/FALSO

1 L'indipendenza permette di scrivere programmi senza conoscere le strutture fisiche dei dati V F
2 L'indipendenza permette di scrivere programmi conoscendo solo lo schema concettuale. V F
3 L'indipendenza permette di formulare interrogazioni senza conoscere le strutture fisiche. V F
4 Dato che i DB sono condivisi aumenta l'efficienza dei programmi che li utilizzano. V F
5 Dato che i DB sono condivisi si riduce la ridondanza e l'inconsistenza. V F
6 Il fatto che le basi di dati siano persistenti ne garantisce l'affidabilità. V F
7 Il fatto che le basi di dati siano persistenti favorisce l'efficienza dei programmi. V F
8 I progettisti della base di dati realizzano il DBMS. V F
9 I progettisti delle applicazioni utilizzano i DB come progettati dal progettista del DBMS. V F

VERIFICA... le conoscenze

COMPLETAMENTO

Completa le frasi seguenti scegliendo tra le parole proposte.

1 Il modello concettuale descrive .... deve essere rappresentato.
cosa – quando – dove – perché – come

2 Il modello logico descrive .... sono organizzati i dati.
cosa – quando – dove – perché – come

3 La modellazione dei .... consiste nella progettazione delle .... del database.
dati – tabelle – schema – relazioni – entità

4 La modellazione funzionale consiste nell'implementazione delle .... e nella creazione delle funzioni che accedono ai dati.
tabelle – relazioni – entità – funzioni – associazioni

5 Nei database la definizione dei dati e i dati stessi sono salvati all'interno ....
del programma – del database – dello schema – dell'oggetto – dell'applicazione

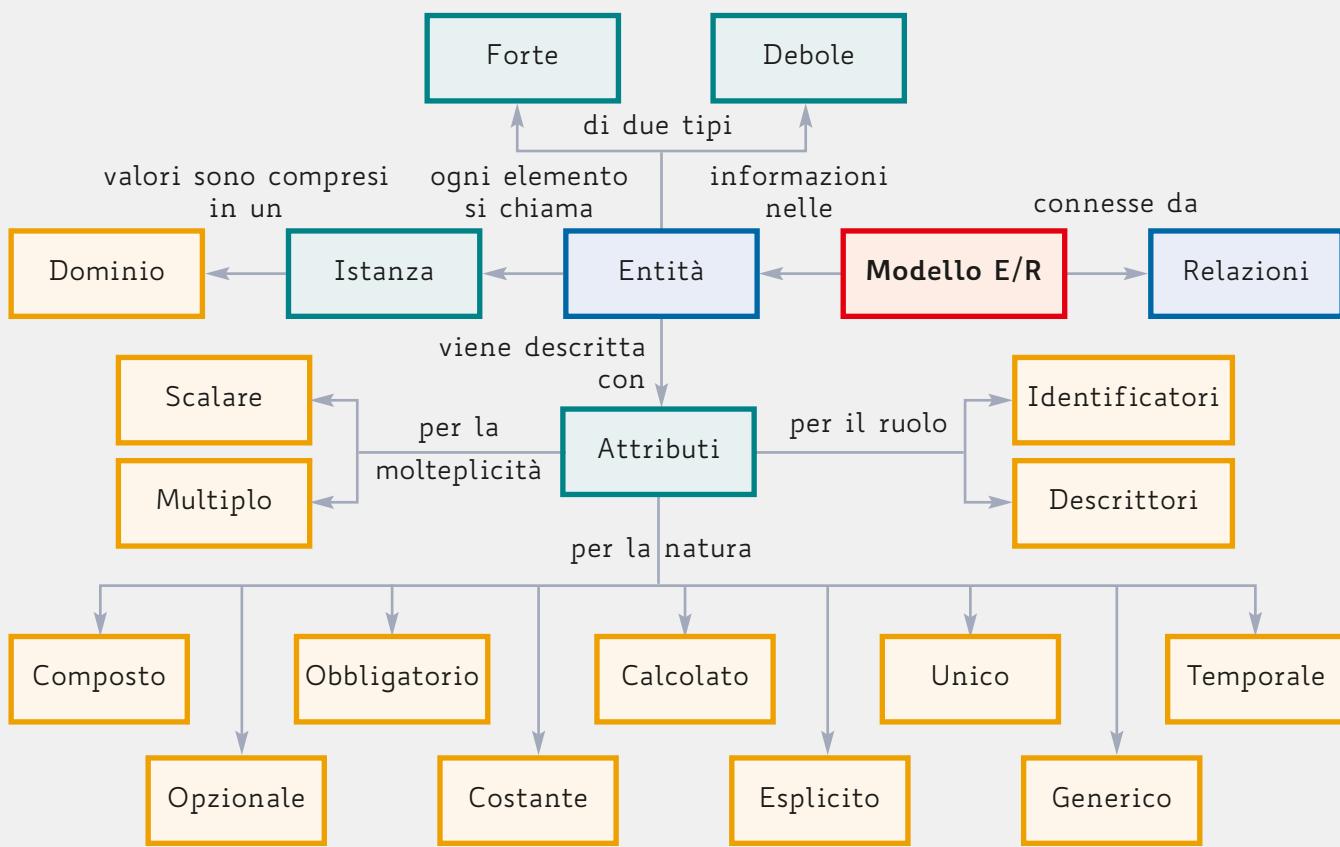
6 Un modello di dati consiste in una rappresentazione astratta delle .... dei dati di un database.
regole – strutture – tipologie – applicazioni – funzioni

7 L'atto di creazione di un modello prende il nome di .... dei dati.
modellazione – rappresentazione – raffigurazione – progettazione

8 Il modello logico deve essere .... dalle strutture fisiche e .... dai programmi applicativi.
corrispondente – dipendente – indipendente – conforme – collegato – utilizzato – modificato – gestito

9 Il modello di database gerarchico prevede che i dati siano organizzati secondo strutture ad ...., che si suppone riflettano una gerarchia esistente tra gli .... che appartengono al database.
file – albero – lista – grafo – puntatore – entità – relazioni – dati – elementi – oggetti

VERIFICA... le competenze

PROBLEMI

1 Rappresenta con il modello gerarchico la situazione Medici – Pazienti.
2 Rappresenta con il modello gerarchico la situazione Squadra – Calciatori.
3 Rappresenta con il modello gerarchico la situazione Alunni – Esami.
4 Rappresenta con il modello gerarchico la situazione Squadra di calcio – Incontri.
5 Rappresenta con il modello reticolare la situazione Fatture – Prodotti.
6 Rappresenta con il modello reticolare la situazione Pazienti – Esami clinici.
7 Rappresenta con il modello reticolare la situazione Automobili – Colori – Marche.

LEZIONE 3

Elementi del modello E-R: entità e attributi

IN QUESTA LEZIONE IMPAREREMO...

- a lavorare con gli elementi principali del modello E-R
- a conoscere il concetto di entità e attributo
- a classificare gli attributi

MAPPA CONCETTUALE

didattica inclusiva

Forte
Debole
di due tipi
valori sono compresi in un
ogni elemento si chiama
informazioni nelle
connesse da
Dominio
Istanza
Entità
Modello E/R
Relazioni
viene descritta con
Scalare
Multiplo
per la molteplicità
Attributi
per il ruolo
Identificatori
Descrittori
per la natura
Composto
Obbligatorio
Calcolato
Unico
Temporale
Opzionale
Costante
Esplicito
Generico

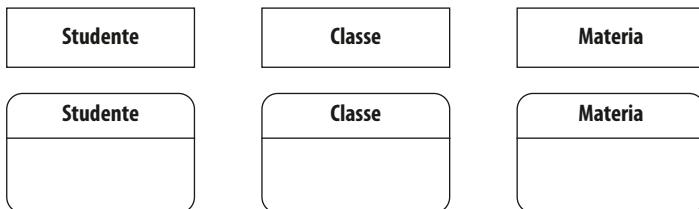
Il modello E-R

Il modello Entità-Relazioni (E-R) è stato originariamente proposto da Chen nel 1976 e aveva come obiettivo principale quello di rendere omogenea la descrizione dei database relazionali in rete.
Concettualmente il modello E-R è piuttosto semplice e il suo nome deriva dal fatto che permette di modellare il mondo reale utilizzando esclusivamente entità e relazioni: il risultato del modello è un diagramma Entità-Relazioni che viene utilizzato per rappresentare visivamente gli oggetti coinvolti nel modello.

L'utilità di questo tipo di modello è così riassumibile:
- i costrutti utilizzati nel modello E-R possono essere facilmente impiegati per la definizione dei database relazionali;
- è semplice e facile da capire con un minimo di guida, perciò il modello può essere usato dai progettisti dei database per comunicarne la struttura agli utenti finali;
- tale modello può essere usato come piano di lavoro per gli sviluppatori del database per implementare un modello di dati in uno specifico software di gestione di database.

Abbiamo detto che il modello E-R permette di modellare graficamente il mondo reale sotto forma di entità e di relazioni tra di esse: in questa lezione esamineremo in che cosa consistono le entità, mentre ci occuperemo delle relazioni nella lezione 5.

Entità

Le entità sono gli oggetti principali su cui vengono raccolte le informazioni.
Ogni entità del modello E-R serve a rappresentare graficamente un concetto, concreto o astratto, del mondo reale (che andiamo a modellare). Un'entità può essere una persona, una macchina, un posto o un evento al quale sono associati dei dati.

Semplificando questa definizione, possiamo dire che un'entità serve per raccogliere informazioni e quindi possiamo pensarla come un oggetto che rappresenta un "gruppo omogeneo di informazioni". A ciascuna entità diamo un nome che ci permette di identificare ogni istanza di quella entità: per esempio, gli studenti di una scuola sono classificabili nell'entità Studente, le materie studiate nell'entità Materia e la classe di appartenenza nell'entità Classe e così via.

Le entità sono "le candidate" a diventare le tabelle del database relazionale.

Le tre entità appena elencate – Studente / Classe / Materia – vengono rappresentate graficamente in questo modo:

oppure:

UML
L'UML (*Unified Modeling Language*, "linguaggio di modellizzazione unificato") è un linguaggio di modellizzazione basato sul paradigma orientato agli oggetti. UML permette, tramite l'utilizzo di modelli visuali, di analizzare, descrivere, specificare e documentare un sistema software anche complesso ed è divenuto, di fatto, uno standard universale adottato nell'ingegneria del software per la descrizione dei progetti.

Non esiste uno standard per rappresentare gli oggetti nei diagrammi E-R, così come esistono svariate metodologie di modellizzazione e ciascuna utilizza formalismi diversi. Alcune linee comuni prevedono tuttavia che le entità vengano rappresentate da rettangoli e le relazioni da linee che collegano i rettangoli.

In questo testo le entità verranno rappresentate o come rettangoli con al centro il nome della entità (primo esempio), come nei più diffusi formalismi dei diagrammi E-R, oppure come rettangoli arrotondati, nei quali l'intestazione rappresenta il nome dell'entità espressa in forma singolare, seguendo la notazione standard del **linguaggio UML** (secondo esempio).

Dai diagrammi E-R si ricavano le tabelle nella fase di progettazione logica e in essa i nomi sono indicati in forma plurale, nel rispetto delle convenzioni adottate universalmente.

A seconda della loro tipologia e delle loro caratteristiche, è possibile distinguere diversi tipi di entità, che esaminiamo nei paragrafi che seguono.

Entità forti ed entità deboli

Un'entità si dice forte se non ha bisogno di altre entità per essere identificata, debole se richiede altre entità per essere identificata.

ESEMPIO

Supponiamo di dover realizzare il sistema informativo di un ospedale. In particolare, gestiamo l'associazione degli *esami* con i *pazienti*. In questo caso, avremo due entità: Paziente ed Esame. Queste due entità sono in relazione tra loro in quanto gli esami vengono effettuati sui pazienti e quindi non possono esistere esami che non sono attribuiti a dei pazienti.
Viceversa, possono esistere dei pazienti a cui non vengono effettuati degli esami.
L'entità Paziente è un'entità forte perché esiste indipendentemente dall'entità Esame; l'entità Esame, invece, è un'entità debole, perché la sua esistenza è giustificata solo dall'esistenza dell'entità Paziente.
Quindi:
- Paziente: entità forte
- Esame: entità debole

Analogamente all'esempio proposto, applichiamo questa distinzione ad altre situazioni.

Autore e Canzone: non esiste *canzone* senza *autore* – Autore: entità forte – Canzone: entità debole	Alunno e Interrogazione: non può esserci *interrogazione* senza *alunno* – Alunno: entità forte – Interrogazione: entità debole	Genitore e Figlio: non possono esserci *figli* senza *genitori* – Genitore: entità forte – Figlio: entità debole

Entità associative

Le entità associative (dette anche entità di intersezione) sono entità usate per associare due o più entità allo scopo di risolvere un'associazione multipla, dove una istanza di una entità viene associata a molteplici istanze di una seconda entità e viceversa.

ESEMPIO

Riprendiamo l'esempio precedente con le due entità Paziente ed Esame. Un paziente fa *molti* esami diversi e un certo esame è eseguito su *molti* pazienti diversi: la relazione è quindi molti a molti e per descrivere correttamente questa situazione è necessario introdurre una nuova entità, il Referto, in modo da collegare (associare) un particolare esame a un singolo paziente.
Referto è un'entità associativa in quanto associa il Paziente all'Esame che ha sostenuto e, per esempio, conterrà la data in cui l'esame è stato effettuato in modo da individuare l'esame desiderato.

Vediamo un secondo esempio.

ESEMPIO

Consideriamo le due entità Docente e Classe: anche in questo caso un Docente insegna in *diverse* classi e una classe ha *diversi* docenti: la relazione è quindi molti-a-molti. È necessario introdurre una nuova entità, l'Orario.
Orario è un'entità associativa in quanto associa il Docente alla Classe.

Le entità associative verranno analizzate anche in seguito, assieme alle relazioni, in quanto serviranno per poter risolvere le relazioni multiple di tipo molti-a-molti (n, n).

Istanze e attributi

Abbiamo visto che l'entità è un elemento (oggetto concreto o astratto) di interesse per la realtà che si vuole modellare: ogni "esemplare" della entità viene chiamata istanza dell'entità ed è caratterizzata da un insieme di valori che descrivono le sue proprietà.

Gli elementi che contengono i dati (valori) dell'istanza sono chiamati attributi (o proprietà).

ESEMPIO

Per l'entità Studente gli attributi possono essere il nome, il cognome, il numero di matricola, l'indirizzo, la classe che frequenta, la sezione ecc.
Una istanza dell'entità Studente, invece, è l'insieme dei dati che caratterizzano un "esemplare" di Studente, per esempio: "*Mario, Rossi, A60900, via Roma, 23, IV info1*".
Una seconda istanza potrebbe essere un secondo esemplare di Studente, con i seguenti valori per gli attributi: "*Giuseppe, Verdi, B6900, via Milano, 12, IV info2*".

Tutte le entità di un dato insieme di entità hanno gli stessi attributi ma ogni istanza ha dei valori diversi per poter essere distinta e individuata tra tutte le altre. Una particolare istanza di un attributo è detta valore.

Un'istanza è quindi l'analogo di un record all'interno di un file, cioè il singolo elemento dell'insieme omogeneo di tutti gli elementi.

Vediamo un secondo esempio.

ESEMPIO

Tracciato record
Il termine tracciato record viene utilizzato per indicare la struttura del record, dove per ogni campo sono descritte le sue caratteristiche, cioè dimensione e tipo (*data-type* e *dimension*).

Consideriamo lo scontrino di acquisto di un generico prodotto in un negozio: prendiamo come esempio il **tracciato record** così costituito:

SCONTRINO		
numero	data	importo

In questo caso abbiamo che:
- Scontrino è l'entità;
- numero, data, importo, cioè i tre dati che lo compongono, sono gli attributi.

Una possibile istanza per ciascun attributo potrebbe invece essere: "123, 10-02-2018, 235.30 Euro".

Classificazione degli attributi

Come abbiamo appena detto, gli attributi (o proprietà) permettono di descrivere le caratteristiche dell'entità alle quali sono associati: equivalgono ai "campi del record".
Nella tabella che segue sono ripotati alcuni semplici esempi di entità con i principali attributi che ne consentono la descrizione e la definizione delle singole istanze.

ENTITÀ	ATTRIBUTI
Studente	Cognome, Nome, Matricola, Anno, Sezione, …
Auto	Targa, Marca, Modello, Colore, Cilindrata, …
Libro	Codice_ISBN, Autori, Titolo, Editore, Prezzo,..
Docente	Cognome, Nome, Telefono, Specializzazione, Materia, Corso, …

Gli attributi hanno particolari caratteristiche che li contraddistinguono e possono essere classificati in base a diverse tassonomie.

Identificatori e descrittori

Una prima classificazione degli attributi li ripartisce in:
- identificatori: chiamati più comunemente chiavi, identificano univocamente un'istanza di un'entità;
- descrittori: descrivono una caratteristica non unica di un'istanza di un'entità.

Gli identificatori permettono dunque di "distinguere" in modo univoco una istanza di una entità, mentre i descrittori completano la descrizione delle caratteristiche dell'istanza, che però possono essere uguali oppure in comune con altre istanze della stessa entità.

Gli attributi chiave, data la loro importanza, saranno dettagliatamente analizzati nella Lezione 4.

Scalari o multipli

Una seconda classificazione degli attributi viene fatta sulla base della molteplicità dell'attributo. Avremo così attributi:
- scalari, cioè semplici, che possono avere un solo valore nell'istanza;
- multipli, cioè semplici, che possono avere più valori (*n* valori) nell'istanza.

ESEMPIO

- Cognome, Nome, Matricola sono attributi scalari.
- Specializzazione, Materia, Autore, Lingua_Parlata sono attributi multipli (o multivalore).

Tipologie di attributi

Una terza classificazione viene fatta in base alla "natura" dell'attributo.

TIPO	DESCRIZIONE	ESEMPI
Semplice (atomico)	Non è ulteriormente scomponibile, elementare	Progressivo, Cognome, Età, Peso
Composto	È costituito da un insieme di componenti.	Telefono, Indirizzo, Data
Opzionale (parziale)	È possibile la sua assenza, cioè potrebbe non esistere in qualche istanza.	Telefono, Coniugato, Interesse
Obbligatorio (totale)	È l'opposto di opzionale e deve sempre essere presente un suo valore in ogni istanza.	Codice_Fiscale, Cognome, Data_Nascita
Costante (statico)	I valori non possono essere cambiati per tutti il "ciclo di vita" dell'attributo.	Codice_Fiscale, Cognome, Nome
Modificabile (dinamico)	È l'opposto di costante, cioè i suoi valori possono venire modificati.	Età, Peso, Indirizzo
Calcolato	Il valore è calcolato con un algoritmo.	Stipendio, Importo_Fattura, Età
Esplicito	È l'opposto di calcolato.	Data_Nascita, Prezzo, Peso
Unico (univoco)	Tutte le istanze della classe hanno valore diverso.	Codice_Fiscale, Partita_IVA
Generico (multivalore)	È l'opposto di unico.	Materie_Insegnate, Lingue_Parlate, Colore
Temporale	Alcuni attributi hanno una validità temporale, cioè dopo un certo tempo non hanno più significatività se non per l'archivio storico.	Stipendio, Giacenza, Prezzo, Mansione, Scadenza

Questa classificazione evidenzia la funzione che gli attributi hanno nelle diverse entità: nei modelli E-R complessi spesso, per semplificare la rappresentazione e migliorare la leggibilità, vengono inseriti solamente gli attributi chiave, necessari per definire le relazioni tra le entità.

Domini

Ogni attributo, essendo sostanzialmente un "contenitore di un dato" assimilabile a una variabile, presenta uno specifico formato: può essere cioè un numero intero, decimale, una stringa di caratteri, un booleano ecc.
In base al loro formato, gli attributi possono assumere un insieme di valori che ne identifica il dominio.

Un dominio è un insieme di valori valido per un attributo e assicura che i valori di un'inserzione o di una variazione abbiano senso: è la collezione di tutti i possibili valori che un attributo può avere.

ESEMPIO

Il dominio dell'attributo nome è una stringa di caratteri, il dominio di numero è l'insieme dei numeri naturali, il dominio di data sono tutti i possibili giorni dalla notte dei tempi alla fine del mondo ecc.
Il dominio dell'attributo mese è costituito dall'insieme {gennaio, febbraio, marzo, ..., dicembre}.

Nella fase di progettazione, per ogni attributo è importante definire un insieme di specifiche che consentono di definirne il dominio e che he elenchiamo di seguito.

- Tipo di dato: i tipi di dato di base sono intero, decimale e carattere. La maggior parte dei database supporta varianti di questi e tipi speciali per data, ora e valori logici.
- Lunghezza: è il numero che indica la quantità di cifre o caratteri usati per rappresentare il valore. Per esempio, un dato di 5 cifre o una parola di 40 caratteri.
- Intervallo: l'intervallo specifica i limiti superiori e inferiori dei valori, e in fase di input queste richieste devono essere soddisfatte.

Tra le specifiche che consentono di definire il dominio di un attributo sono inoltre da tenere in considerazione anche le informazioni su due situazioni iniziali di particolare interesse operativo:

- supporto dei valori null: il valore null è quello che indica che al campo non è assegnato alcun valore.
- valore di default: il valore di un'istanza assume questo valore se non viene specificato altrimenti.

Vediamo un esempio riepilogativo, in cui viene fornita la descrizione completa degli attributi di una entità e che potremo prendere come "struttura di riferimento base" di schema logico: nelle prossime lezioni lo integreremo con ulteriori informazioni che ci consentiranno di arrivare alla definizione delle tabelle dei database.

ESEMPIO

Riportiamo alcuni attributi dell'entità Persona indicando vicino a essi le specifiche sopra indicate, che potrebbero essere le seguenti:

- nome: stringa(20) – obbligatorio - non NULL
- cognome: stringa(20) – obbligatorio - non NULL
- cod_fiscale: stringa(16)
- titolo_di_studio: stringa(50)

– data_di_nascita: giorno - mese – anno
– peso: numerico (6,2)
– anzianità_servizio: numerico(6)

In questo esempio non è necessario indicare i domini in quanto per gli attributi di tipo stringa questi sono gli intervalli (a..Z, 0..1), mentre per gli attributi giorno, mese e anno – componenti di data_di_nascita – sono:

– giorno = 1, ..., 31
– mese = {Gen, Feb, Mar, Apr, Mag, Giu, Lug, Ago, Set, Ott, Nov, Dic}
– anno = 1900, ..., 2100

Vincoli

Non vanno inoltre trascurate le specifiche su eventuali vincoli, cioè su speciali restrizioni sui valori ammessi; al loro inserimento e durante la loro creazione i valori devono essere controllati sulla base di vincoli definiti in sede di analisi.
I vincoli non vengono rappresentati graficamente nei diagrammi E-R ma sono descritti in specifiche a parte che devono essere tenute in considerazione nella realizzazione del software, in quanto è necessario che questo effettui sempre il controllo dell'appartenenza di un valore di un attributo a un preciso dominio.

I vincoli possono essere di due tipi: statici e dinamici.

I vincoli statici richiedono che il dato venga controllato solo al momento dell'inserimento e/o della sua variazione, e sono:
– vincoli di correttezza in caso che il dato sia un codice composto verificabile con algoritmi, come il codice fiscale;
– presenza del valore in elenchi predefiniti (che prendono il nome di dizionari);
– vincolo sul valore del dato dipendente da valori di altri attributi presenti nel DB.

ESEMPIO

Consideriamo il caso di una entità Alunno e di due dei suoi attributi: come vincolo su nome e cognome si potrebbe mettere che, essendo stringhe alfabetiche, non devono contenere numeri, digitati per errore:

Giuseppe	**SÌ**	G1useppe	**NO**
Mario	**SÌ**	Mari0	**NO**

Oppure si potrebbe richiedere, nel caso di un attributo contenente una parola composta, che non siano presenti elementi di punteggiatura separatori indesiderati (come "/" oppure "-"):

De Gennaro **SÌ** De-Gennaro **NO**

Anche per il peso si potrebbe inserire un vincolo, in modo da verificare la coerenza dei valori che verranno inseriti:

0,2 < peso <200,0

Possiamo anche individuare vincoli statici "dipendenti", come nella seguente coppia di attributi:

Anno_Di_Corso: numerico
Esame_Maturità: booleano, vincolato da Anno_Di_Corso = 5

Il valore del secondo attributo sarà VERO solo se il valore del primo è uguale a 5.

I vincoli dinamici necessitano di un controllo periodico in quanto il valore può uscire dal dominio di definizione durante l'esercizio del sistema a causa di eventi che fanno cambiare lo stato di un attributo e/o semplicemente per effetto della variabile "tempo".

– un Prodotto scade dopo una certa data
– un Libro è una Novità entro 14 giorni dal lancio commerciale
– lo Stipendio_Base precedente ≤ Stipendio_Base successivo
– Età precedente ≤ Età successiva

ESEMPIO

Completiamo l'esempio riepilogativo sull'entità Persona aggiungendo i vincoli su alcuni attributi:
– cod_fiscale: attributo calcolato da altri attributi (statico)
– titolo_di_studio: descrizione presente in un elenco (statico)
– anzianità_servizio: maggiore del valore precedente (dinamico)

Validazione degli attributi

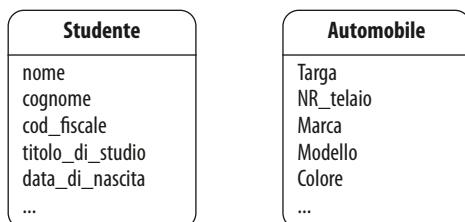
Premesso che tutti i valori assegnati agli attributi devono essere sempre validati, i controlli devono essere effettuati:
– sia in fase di caricamento dati che in fase di aggiornamento;
– in fase di interrogazione;
– periodicamente per quegli attributi che hanno vincoli dinamici.

La necessità di validazione di dati si rende necessaria anche a causa di errori accidentali legati all'inserimento manuale dei dati stessi: dove possibile è consigliabile l'utilizzo di sistemi hardware di riconoscimento dei dati, quali codici a barre, carte magnetiche ed elettroniche, QR Code.

Inclusione degli attributi nel diagramma E-R

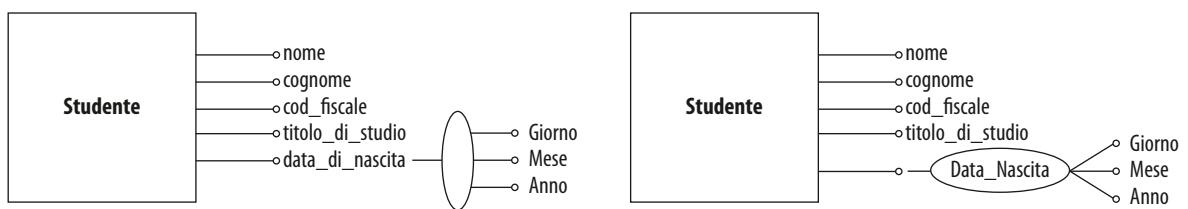
Ricapitolando, per ogni attributo è sempre necessario individuare tre caratteristiche fondamentali:
– il formato, che indica il tipo di valori che assume;
– la dimensione, che è un numero che indica la quantità massima di caratteri o cifre inseribili e il suo dominio;
– l'opzionalità, che indica la possibilità di non essere sempre valorizzato:
  • l'attributo è obbligatorio se deve avere un valore non nullo (per esempio il nome di una persona);
  • l'attributo è facoltativo se sono accettabili valori nulli (per esempio il titolo di studio di una persona).

C'è disaccordo sul fatto che gli attributi debbano far parte o meno del diagramma E-R: secondo alcuni progettisti gli attributi dovrebbero essere tutti presenti, mentre altri si limitano a inserire quelli più significativi.

Alcuni esperti, comunque, hanno osservato che aggiungere gli attributi, specialmente dove ce ne sono molti, spezza il diagramma impedendo di presentare all'utente finale una visione della struttura dei dati.

Noi adotteremo un comportamento intermedio, indicando solo le chiavi ed eventualmente uno o due attributi più significativi nei casi in cui tali attributi possano essere utili per una migliore comprensione dello schema.

Notazione grafica per gli attributi

Nei diagrammi E-R non è obbligatorio riportare tutti gli attributi: come vedremo sarà sufficiente indicare solo quelli più significativi, e per essi viene utilizzata la seguente grafica.

Nel caso si utilizzi la notazione UML avremo:

Gli attributi composti, come Telefono, Indirizzo, Data_Nascita, vengono rappresentati secondo una delle due notazioni grafiche di seguito riportate.

ESEMPIO

Completiamo il diagramma dello Studente, dove è presente l'attributo composto data_di_nascita che possiede tre sotto-attributi (Anno, Mese, Giorno).

Nella notazione UML non è prevista la rappresentazione degli attributi composti.

VERIFICA... le conoscenze

ASSOCIAZIONE

1 Associa ciascuno dei seguenti attributi alla tipologia corretta (scalari o multipli).

Codice_Fiscale	S M	Colore_Occhi	S M	Anzianità	S M
Colore	S M	Libro	S M	Rata mutuo	S M
Patente	S M	Stipendio_Base	S M	Professione	S M
Partita_IVA	S M	Età precedente	S M	Giornale	S M
Rivista	S M	Titolo di studio	S M	Parentela	S M

2 Associa ciascuno dei seguenti attributi alla tipologia corretta (semplici o composti).

Codice_Fiscale	S C	Rata_mutuo	S C	Tasse	S C
Data_Nascita	S C	Città_Abitazione	S C	Professione	S C
Patente	S C	Stipendio_Base	S C	Giornale	S C
Partita_IVA	S C	Età precedente	S C	Referto	S C
Modello	S C	Titolo di studio	S C	Parentela	S C

3 Associa ciascuno dei seguenti attributi alla tipologia corretta (statici o dinamici).

Codice_Fiscale	S D	Colore_Occhi	S D	Anzianità	S D
Prodotto_Alimentare	S D	Libro	S D	Professione	S D
Patente	S D	Stipendio_Base	S D	Giornale	S D
Partita_IVA	S D	Età precedente	S D	Grado (esercito)	S D
Rivista	S D	Titolo di studio	S D	Parentela	S D

4 Associa ciascuno dei seguenti attributi alla tipologia corretta (esplicito o calcolato).

Codice_Fiscale	E C	Imponibile	E C	Anzianità	E C
Prodotto_Alimentare	E C	Libro	E C	Professione	E C
Validità patente	E C	Stipendio	E C	Prezzo	E C
Aliquota_IVA	E C	Età precedente	E C	Sconto	E C
Peso	E C	Interessi bancari	E C	Parentela	E C

5 Associa ciascuno dei seguenti attributi alla tipologia corretta (permanente o temporaneo).

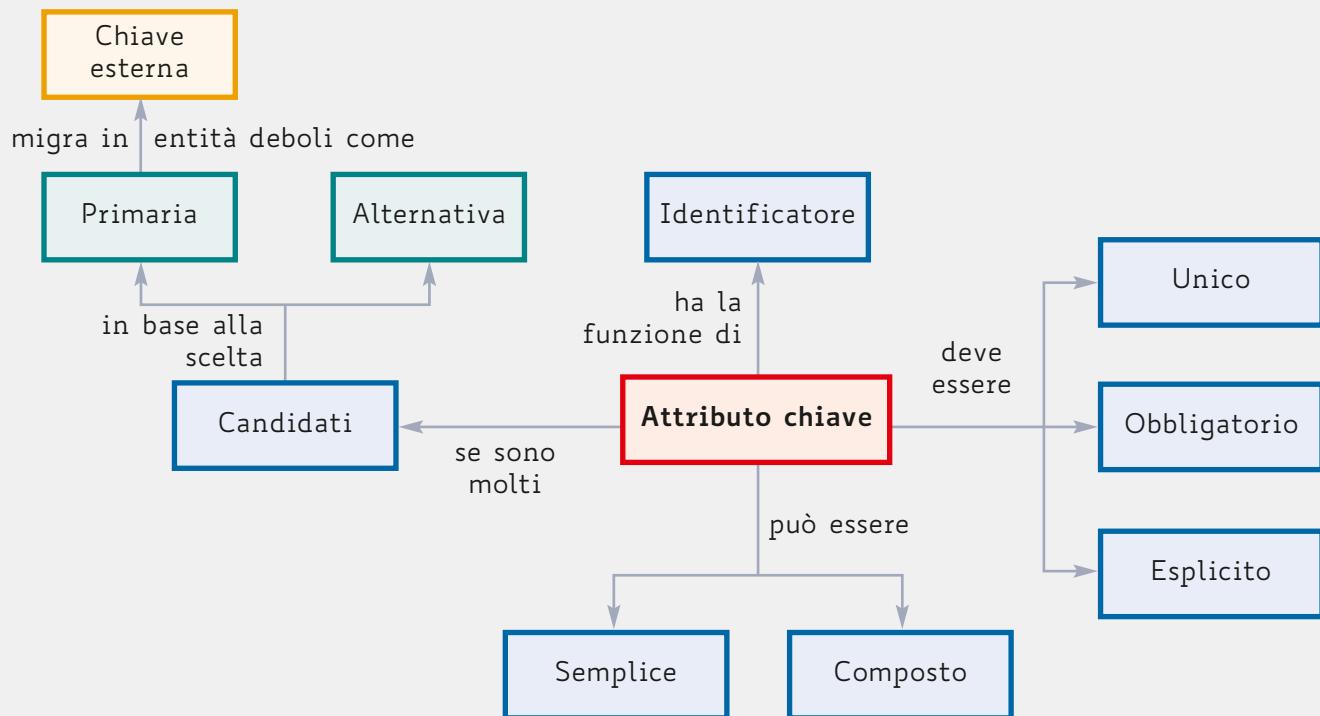
Codice_Fiscale	P T	Imponibile	P T	Anzianità	P T
Quantità in magazzino	P T	Titolo libro	P T	Professione	P T
Nr. patente	P T	Stipendio	P T	Giornale	P T
Partita IVA	P T	Fatturato annuo	P T	Grado (esercito)	P T
Prezzo	P T	Titolo di studio	P T	CAP	P T

VERIFICA... le conoscenze

VERO/FALSO

1 Un modello di dati consiste in una rappresentazione astratta delle strutture dei dati. V F
2 Il modello E-R permette di modellare graficamente il mondo reale. V F
3 Un'entità si dice debole se non ha bisogno di altre entità per essere identificata. V F
4 Le entità associative sono anche dette entità di intersezione. V F
5 Le entità descrivono gli attributi con i quali sono associate. V F
6 Il dominio di un attributo è la collezione di tutti i valori possibili che un attributo può avere. V F
7 Un descrittore descrive una caratteristica unica di un'istanza di un'entità. V F
8 Un attributo è obbligatorio se deve sempre essere presente un suo valore in ogni istanza. V F
9 Gli attributi unici in tutte le istanze della classe hanno valore diverso. V F
10 Gli attributi temporali dopo un certo tempo non hanno più significatività nell'archivio storico. V F

VERIFICA... le competenze

Classifica i seguenti attributi: indica se sono (S) o no (N) identificatori, scalari e riportane la tipologia

ATTRIBUTO	IDENTIFICATORE	SCALARE	TIPOLOGIA
Marca	S N	S N	
Cognome	S N	S N	
Telefono	S N	S N	
Partita iva	S N	S N	
Stipendio	S N	S N	
Cittadinanza	S N	S N	
Materia insegnata	S N	S N	
Data nascita	S N	S N	
Colore	S N	S N	
Età	S N	S N	
Data	S N	S N	
Prezzo	S N	S N	
Indirizzo	S N	S N	
Coniugato	S N	S N	
Nome	S N	S N	
Peso	S N	S N	
Scadenza	S N	S N	
Codice fiscale	S N	S N	
Importo fattura	S N	S N	
Lingue parlate	S N	S N	

Elementi del modello E-R: gli attributi chiave

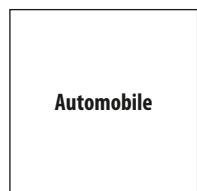
IN QUESTA LEZIONE IMPAREREMO...

- a individuare le chiavi primarie
- ad aggiungere gli attributi al modello E-R

MAPPA CONCETTUALE

Attributi chiave-identificatori

Nella precedente lezione abbiamo detto che gli attributi chiave sono gli identificatori delle istanze, ci permettono cioè di "distinguere" in modo univoco un'istanza di una certa entità.

Un attributo chiave deve avere le seguenti caratteristiche:
– deve essere obbligatorio, unico, esplicito;
– può essere composto e nessun attributo può avere valore null;
– non è modificabile.

ESEMPIO

Automobile

<u>Targa</u>
NR_telaio
Marca
Modello
...

Riportiamo gli attributi di un'entità Automobile rappresentandola secondo la notazione UML. Individuiamo come chiave l'attributo Targa, che svolge un ruolo diverso dagli altri attributi, in quanto ci permette di individuare univocamente una specifica automobile.

Nello schema UML abbiamo sottolineato questo attributo, così da differenziarlo dagli altri che descrivono la nostra automobile, e lo abbiamo inserito all'interno della entità.

Possiamo anche rappresentare la medesima entità con la notazione classica del modello E-R, cioè a rettangoli-rombi: in questa notazione gli attributi sono indicati all'esterno dell'entità e collegati con una linea terminante con un pallino:

– nero pieno per la chiave;
– bianco per gli altri attributi.

In una entità possono anche essere presenti più attributi con caratteristiche tali da renderli possibili campi chiave: ogni chiave potenziale prende il nome di chiave candidata.

Una volta che le chiavi candidate sono state identificate è necessario sceglierne una sola, che verrà utilizzata come "chiave di riferimento" per quella entità: questa chiave prende nome di chiave primaria, mentre alle altre viene dato il nome di chiavi alternative.

La chiave primaria è un attributo o un insieme di attributi che identifica univocamente una specifica istanza di un'entità.
Le chiavi candidate che non sono scelte come chiavi primarie sono dette chiavi alternative.

Automobile

Targa (pk)
NR_telaio
Marca
Modello
...

In ogni entità è dunque necessario definire una sola chiave primaria: negli schemi relazionali essa viene indicata secondo le convenzioni grafiche riportate nell'esempio sopra proposto; in alternativa, nella grafica UML, al posto della sottolineatura può essere indicata semplicemente aggiungendo la sigla (pk), iniziali di *primary key*.
In sintesi, un attributo, per poter essere la chiave primaria di un'entità, deve avere le seguenti caratteristiche e proprietà:

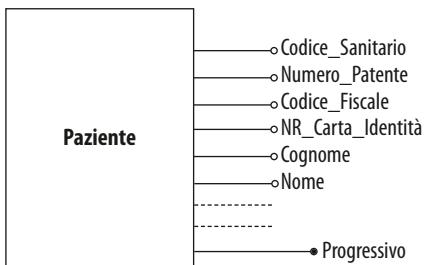
– il suo valore deve essere specificato per ogni istanza dell'entità;
– il valore deve essere unico per ogni istanza dell'entità;
– il valore non deve cambiare durante la vita di ogni istanza dell'entità.

ESEMPIO

Indichiamo gli attributi di un'entità Impiegato e ne individuiamo due che possono essere candidati a essere attributi chiave: il Codice_Fiscale e il Cognome + Nome. Il Codice_Fiscale è sicuramente unico per tutti i cittadini italiani, mentre Cognome + Nome non garantiscono l'univoca individuazione della persona perché sono possibili omonimie: viene quindi scelto come attributo chiave il Codice_Fiscale.
Rappresentiamo ora l'entità:

con la grafica UML

Chiavi artificiali

In alcune situazioni si rivela necessario aggiungere un attributo che abbia garanzie di sicurezza e rispetti le condizioni per essere utilizzato come chiave primaria: si tratta di un attributo numerico, che assume valori progressivi assegnati dal sistema garantendo in tale modo l'unicità e che prende il nome di chiave artificiale (o serial).

Una chiave artificiale è formata da un attributo privo di significato proprio che viene aggiunto agli altri in modo strumentale per ottenere un codice univoco per ogni istanza: è composta da un contatore che si incrementa automaticamente (*autoincrement*).

La scelta di introdurre una chiave artificiale viene effettuata quando:
– nessun attributo ha tutte le proprietà di una chiave primaria;
– le chiavi candidate possono creare problemi nel tempo;
– la chiave primaria è grande e complessa;
– ci sono problemi di identificazione.

ESEMPIO

Un tipico esempio dove viene utilizzata una chiave artificiale per problemi di identificazione è quello del Pronto Soccorso: nonostante un paziente abbia più chiavi candidate, viene riconosciuto in base all'ordine di accettazione (Progressivo), e questo codice diviene la chiave primaria dell'entità Paziente.
Tutti i primi attributi sono chiavi candidate, ma la scelta cade sulla chiave artificiale.

Campo di tipo autoincrement
Il tipo di campo di tipo autoincrement, oltre che in Access, è presente in molti DBMS, tra i quali ricordiamo MySQL, SQL Server, ma non è presente in Oracle: deve essere realizzato mediante un tricky, cioè un trucco, aggiungendo alcune righe di codice (`MINVALUE 1,START WITH 1, INCREMENT BY 1`) che vengono richiamate all'inserimento di un nuovo elemento.

La maggior parte dei DMBS ammettono come tipo base proprio il tipo contatore e agli attributi definiti come contatore assegnano automaticamente il valore incrementandolo di 1 per ogni record che viene aggiunto (**campo di tipo autoincrement**).

A questo campo viene convenzionalmente dato un nome che inizia con ID (identificatore) seguito dal nome dell'entità (per esempio ID_Alunno, ID_Auto, ID_Citta ecc.).

ESEMPIO

Anche nella vita quotidiana si fa sovente ricorso a degli attributi numerici che sono, di fatto, delle chiavi artificiali, in quanto seguono una numerazione progressiva, come per esempio il numero di matricola degli alunni, la partita IVA, il numero di telaio di una automobile ecc.

Scelta della chiave e del codice univoco

Non sempre è semplice scegliere la chiave primaria in presenza di molte chiavi candidate e spesso la scelta viene effettuata analizzando un insieme di fattori "esterni" al progetto del database.

ESEMPIO

Esaminiamo il caso della entità Studenti: è preferibile l'utilizzo di Codice_Fiscale oppure di Numero_Matricola come chiave primaria?
Dal punto di vista della velocità di digitazione è senza dubbio preferibile il numero di matricola, ma in termini di sicurezza il codice fiscale, grazie alla cifra di parità, ci permette di verificare l'esattezza della sua digitazione.

A volte viene invece scelto come chiave primaria un attributo che è già stato usato come identificatore da parte di altre organizzazioni e/o applicazioni.

ESEMPIO

Possiamo utilizzare come chiave primaria degli articoli di magazzino il codice che utilizza il nostro fornitore, oppure la partita IVA per individuare i nostri fornitori o il codice fiscale per i nostri clienti o, infine, il CAP per individuare il comune di residenza.

Questa scelta, "apparentemente felice", può invece rivelarsi pericolosa in quanto, di fatto, si diviene "dipendenti" dalle scelte di altre organizzazioni che possono mutare nel tempo... a nostra insaputa!

La scelta della chiave primaria potrebbe anche ricadere sui "vecchi" codici parlanti presenti nella maggior parte delle aziende manifatturiere: si tratta di codici che venivano attribuiti per riconoscere direttamente la tipologia del prodotto in quanto "racchiudevano "molte informazioni in un'unica stringa di caratteri.

ESEMPIO

Sono esempi di codici parlanti:
- Codice_Inventario: ST 422 (specie, numero)
- Codice_Materiale: GMN 22 specie, diametro)
- Codice_Articolo: PNT RS 40 (specie, colore, taglia)
- Numero_Matricola: 04 02 7343 (indirizzo, specializzazione, numero)
- Protocollo: 03 PT 144 (ufficio, iniziali, numero)

L'utilizzo di tali codici come chiave primaria è assolutamente da evitare.

I codici parlanti possono infatti creare una serie di problemi:
- nel caso per esempio del Numero_Matricola, se un alunno cambia specializzazione si "dovrebbe" cambiare il codice per mantenerne la significatività, ma in questo modo si perde la sua storia pregressa a meno di non "portarsi" dietro sempre due codici;
- anche nel caso del Codice_Inventario, dove il codice è composto da una specie e un numero, se si introducono nuove classificazioni con nuove abbreviazioni, si possono verificare anomalie.

ESEMPIO

Se si decide si "specializzare" il codice dei personal computer (che "naturalmente" è costituito dalle iniziali PC) per differenziare le tipologie in:
- PCD, per i PC di tipo desktop;
- PCT, per i PC di tipo tower;
- PCP, per i PC di tipo portatile

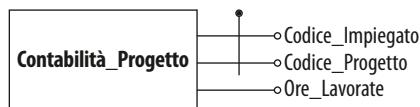
È necessario effettuare delle sostituzioni alle istanze già presenti in archivio e questo comporta la necessità di revisionare l'intero dell'archivio con enorme dispendio di tempo!

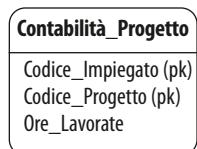
Chiavi composte

In alcuni casi sono necessari più attributi per identificare univocamente un'entità.

Una chiave primaria formata da più di un attributo viene detta chiave composta.

ESEMPIO

Contabilità_Progetto

Vogliamo definire un'entità Contabilità_Progetto relativa alle ore dedicate da ciascun impiegato ai diversi progetti in cui è impegnato. I dati relativi all'entità Contabilità_Progetto possono essere raccolti in una tabella come quella riportata di seguito.

CONTABILITÀ_PROGETTO		
Codice_Impiegato	**Codice_Progetto**	**Ore_Lavorate**
01	01	200
01	02	120
02	01	50
02	03	120
03	03	100
03	04	200

Nella tabella si può notare che il Codice_Impiegato è ripetuto su più righe e, quindi, non può essere scelto come chiave primaria; analoga osservazione può esser fatta per il Codice_Progetto. Solo l'abbinamento dei due codici permette di creare un dato univoco per ciascuna riga.

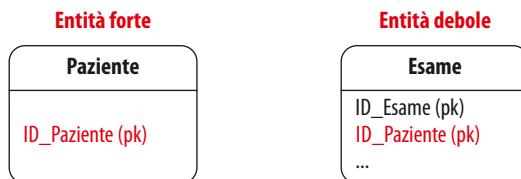
Si ottiene così una chiave composta da due attributi: Codice_Impiegato + Codice_Progetto.

La rappresentazione grafica E-R è la seguente.

Nello grafica UML questa situazione viene invece indicata riportando entrambi gli attributi con l'indicazione di pk.

Nella tabella sotto riportata vengono illustrate quattro possibili situazioni in cui sono presenti uno o più identificatori.

Migrazione di chiave primaria

La necessità di introdurre chiavi composte si presenta per esempio nel caso di entità deboli (o dipendenti), ovvero di entità che dipendono dall'esistenza di un'altra entità per la loro identificazione.

Gli elementi di queste entità sono identificati dall'insieme di due attributi, uno dei quali è sempre la chiave primaria della entità forte (padre).

Le entità deboli (figli) ereditano la chiave primaria della corrispondente entità forte (padre): si dice che viene effettuata una migrazione della chiave primaria da padre a figlio.

ESEMPIO

Nella situazione Esame (del sangue)-Paziente è evidente che senza il paziente non esistono esami, quindi Paziente è l'entità forte e la sua chiave primaria migrerà nella entità debole Esame.

La chiave primaria dell'entità Esame risulta essere formata dall'insieme delle due chiavi, è quindi una chiave composta dall'identificatore del Paziente e da quello dell'Esame.

Vediamo un secondo esempio in ambito scolastico.

ESEMPIO

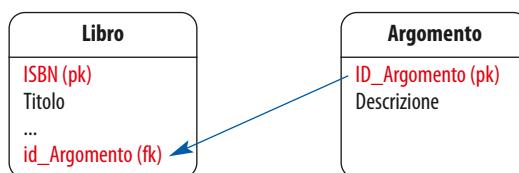
Si vuole rappresentare la situazione delle interrogazioni sostenute da un alunno nelle diverse materie scolastiche: individuiamo come entità forte Alunno e come entità debole Interrogazione.

Anche in questo caso, la chiave dell'entità Interrogazione risulta essere formata dall'insieme delle due chiavi, è quindi una chiave composta dall'identificatore dell'Alunno e da quello della Materia.

Vedremo in seguito come questa situazione risulti più complessa di come è stata presentata in quanto, per completezza, si dovrebbero introdurre come chiave anche altri attributi, come per esempio Data/Ora, per rendere unica l'identificazione.

Chiave esterna

Non sempre l'attributo che migra della entità padre alla entità figlio diviene una "componente" della chiave primaria: in molte situazioni, questo attributo prende il nome di chiave esterna e gioca un ruolo essenziale nella descrizione delle relazioni nel modello E-R in quanto è tramite essa che vengono create le connessioni tra le diverse entità.

Una chiave esterna è un attributo che completa una relazione attraverso l'identificazione dell'entità padre.

Le chiavi esterne forniscono un metodo per mantenere l'integrità dei dati (chiamata integrità referenziale) e per "navigare" tra diverse istanze di un'entità: come vedremo in seguito, ogni relazione 1-a-molti presente nello schema relazionale deve essere supportata da una chiave esterna.

L'attributo chiave esterna viene indicato nello schema relazionale con la notazione (fk), iniziali di *foreign key*, oppure con l'aggiunta di un asterisco *.

ESEMPIO

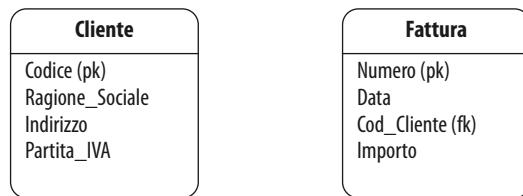
Vediamo una semplice situazione dove sono presenti l'entità Libro e l'entità Argomento, attraverso le quali classificare i diversi volumi.
Le chiavi primarie per le due entità sono per esempio il codice ISBN per il libro e ID_Argomento per l'argomento: la classificazione dei libri avviene introducendo in ogni libro un attributo che permette di sapere l'argomento da esso trattato, e questo attributo è la chiave esterna migrata dalla entità Argomento.

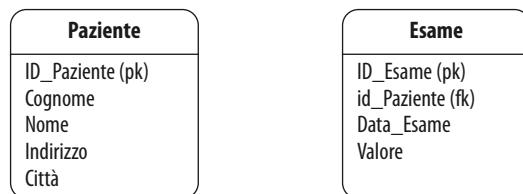
Assegnazione dei nomi alle chiavi esterne

Nel caso in cui le chiavi esterne siano costituite da chiavi artificiali, la convenzione adottata per nominarle è simile a quella introdotta precedentemente per le chiavi primarie: si assegna cioè lo stesso nome preceduto da un id, minuscolo in questo caso.
Vediamo alcuni esempi.

CHIAVE PRIMARIA	CHIAVE ESTERNA
ID_Alunno	id_Alunno
ID_Prodotto	id_Prodotto
ID_Libro	id_Libro
ID_Materia	id_Materia
ID_Nazione	id_Nazione

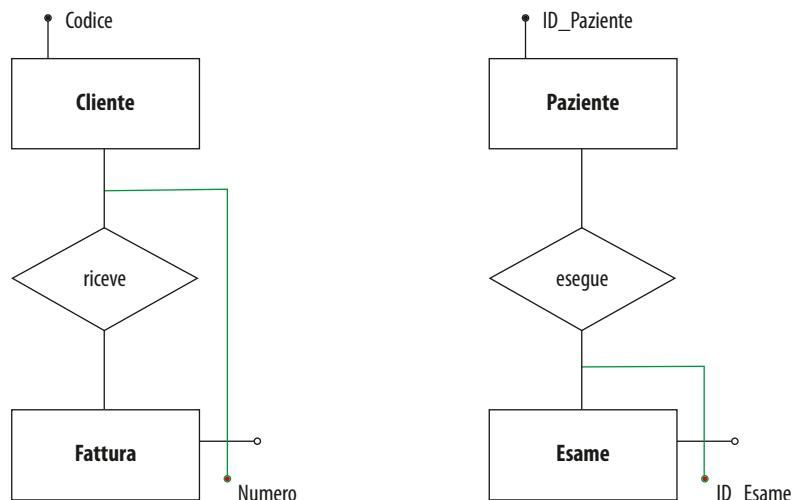
Se l'entità padre ha invece una chiave primaria che è un generico codice (ma non una chiave artificiale), la convenzione adottata dalla maggior parte dei progettisti per nominarla è quella di far precedere, ove possibile, il prefisso cod alla chiave primaria, per migliorarne la leggibilità.
Riportiamo alcuni esempi nella seguente tabella.

CHIAVE PRIMARIA	CHIAVE ESTERNA
Cliente	cod_Cliente
Fornitore	cod_Fornitore
Prodotto	cod_Prodotto
Medico	cod_Medico
Paziente	cod_Paziente

Nel caso in cui la chiave primaria venga indicata semplicemente con il nome di codice nell'entità forte, il nome della chiave esterna si ottiene aggiungendo a cod il nome al singolare dell'entità stessa.

CHIAVE PRIMARIA	NOME ENTITÀ	CHIAVE ESTERNA
codice	Cliente	cod_Cliente
codice	Fornitore	cod_Fornitore
codice	Prodotto	cod_Prodotto
codice	Medico	cod_Medico
codice	Paziente	cod_Paziente

ESEMPIO

Vediamo due esempi: entità Cliente e Fattura e entità Paziente ed Esame.

Nel primo caso abbiamo:

Cliente
Codice (pk)
Ragione_Sociale
Indirizzo
Partita_IVA

Fattura
Numero (pk)
Data
Cod_Cliente (fk)
Importo

Nella seconda situazione si ha:

Paziente
ID_Paziente (pk)
Cognome
Nome
Indirizzo
Città

Nei diagrammi E-R è preferibile indicare la presenza di chiavi esterne sulle entità deboli riportando una “spezzata”, indicata in verde nel disegno seguente, sulla chiave primaria.

La spezzata può essere collocata indifferentemente “sopra” oppure “sotto” il rombo che, come vedremo in seguito, è il simbolo che indica la relazione.

Schema relazionale

L'entità può essere rappresentata, oltre che con lo schema grafico, anche mediante una scrittura che prende il nome di schema relazionale: in alcune trattazioni lo schema relazionale viene indicato come schema logico semplice o semplificato.

Mentre nello schema concettuale non esiste una convenzione comune su come indicare il nome dell'entità (generalmente si utilizza il sostantivo al singolare anche se alcuni teorici lo indicano al plurale), nello schema relazionale, invece, il nome dell'entità viene riportato al plurale, in quanto corrisponderà al nome della tabella del database che, convenzionalmente, deve sempre essere indicata al plurale.

Lo schema relazionale di una tabella può assumere diverse forme. Le tre più utilizzate sono:

1 Impiegati (CodFiscale, Nome, Telefono)

dove viene sottolineato l'attributo che è la chiave primaria.

2 Impiegati (CodFiscale (pk), Nome, Telefono, Nazione (fk))

dove a fianco degli attributi chiave vengono riportate le sigle (pk) e (fk)

3 Impiegati (**CodFiscale**, Nome, Telefono, Città*)

dove viene riportato in grassetto l'attributo che è la chiave primaria e aggiunto l'asterisco (*) alla chiave esterna.

È possibile anche aggiungere, per i singoli campi, alcune informazioni di dettaglio che rendono questa rappresentazione più accurata, come per esempio.

4 Impiegati (**CodFiscale**: string(16), Nome: string(30), Telefono: optional string(30))

Nella nostra trattazione noi utilizzeremo la seconda notazione, cioè quella con la chiave primaria indicata con la sigla (pk).

L'obiettivo della progettazione di un database è quello di definire le tabelle nel DBMS relazionale e, quindi, più dettagli sono presenti nello schema logico e più correttamente e semplicemente verranno definite le singole tabelle e solo l'ultima formulazione dello schema relazionale è sufficientemente esaustiva per assolvere tale compito.
In questo corso adotteremo uno schema logico più completo possibile, "arricchito " con le informazioni sulla tipologia, dominio, caratteristiche, ecc. per ciascun attributo in modo che possa essere di supporto sia par la sua implementazione fisica che per il successivo sviluppo delle applicazioni.

VERIFICA... le conoscenze

VERO/FALSO

1 Le chiavi primarie rafforzano l'integrità dell'entità identificandone univocamente le istanze. V F
2 Le chiavi esterne rafforzano l'integrità referenziale completando l'associazione tra due entità. V F
3 Il valore delle chiavi esterne deve essere unico per ogni istanza dell'entità. V F
4 Ogni chiave che può essere chiave primaria viene chiamata chiave candidata. V F
5 Una chiave primaria che è formata da più di un attributo viene detta chiave composta. V F
6 Una chiave artificiale è formata da un attributo privo di un significato proprio. V F
7 Un'entità viene chiamata entità associativa se associa un padre con un figlio. V F
8 Il valore delle chiavi primarie deve essere unico ed è ammesso un solo valore null. V F

VERIFICA... le competenze

ESERCIZI

Indica le chiavi primarie nelle seguenti entità.

Persona

Abitazione

Autore

Docente

Cantante

Canzone

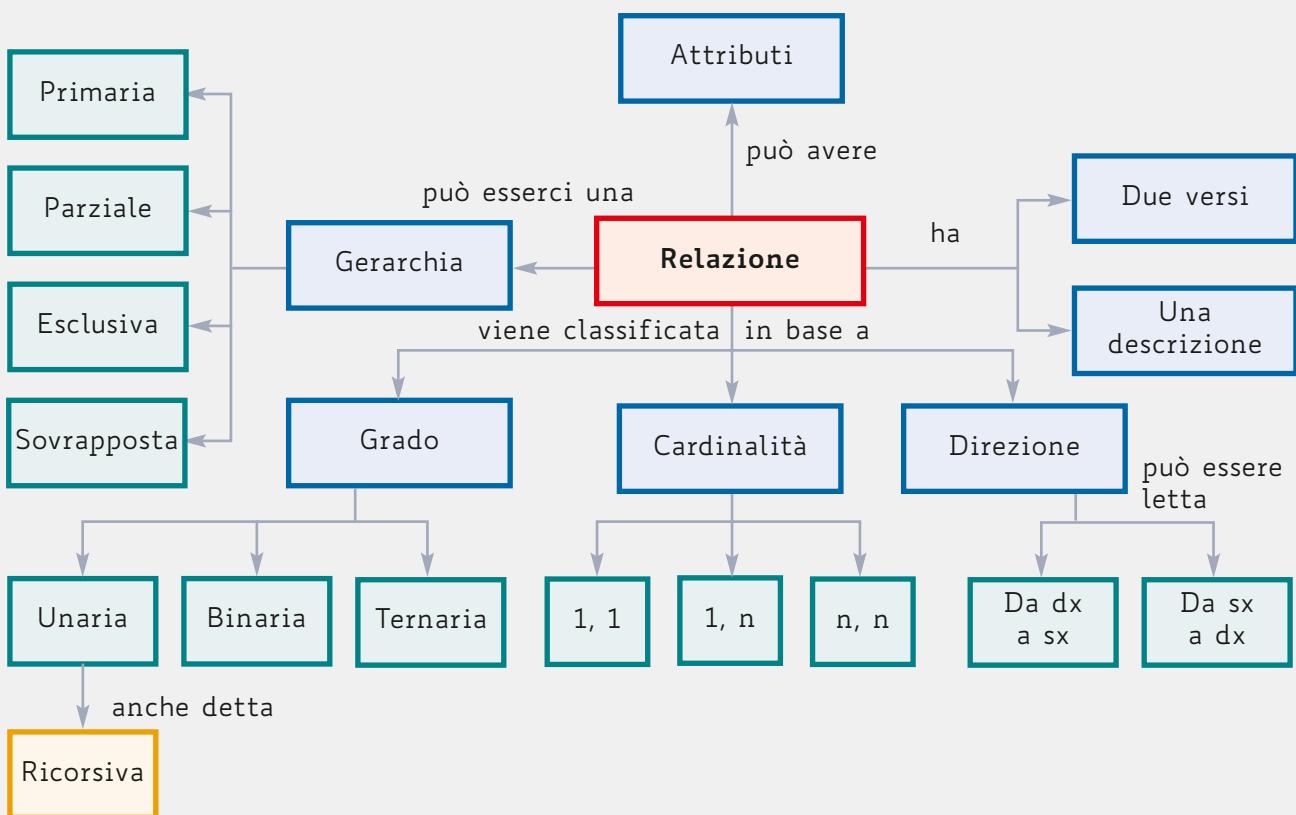
Multa

Automobilista

Verifica di fine Lezione

VERIFICA... le competenze

PROBLEMI

Nelle seguenti situazioni, dopo aver individuato le entità, aggiungi gli attributi e indica le possibili chiavi.

1. Si devono gestire i dati relativi a un campionato automobilistico: è necessario registrare le informazioni relative ai piloti, alle case automobilistiche e ai risultati di ogni gara.

2. Si devono gestire i dati relativi alle assenze degli alunni di una scuola: è necessario registrare le informazioni relative agli studenti e alle assenze.

3. Si devono gestire i dati relativi al rendimento scolastico degli studenti di una classe in una materia: è necessario registrare le informazioni relative agli studenti e ai voti.

4. Si devono gestire i dati relativi al rendimento scolastico degli studenti di una classe in tutte le materie: è necessario registrare le informazioni relative agli studenti e ai voti nelle varie discipline.

5. Si devono gestire i dati relativi a una compagnia di assicurazioni: è necessario registrare le informazioni relative ai clienti, alle auto e alle assicurazioni stipulate.

6. Si devono gestire i dati relativi a una squadra di calcio: è necessario registrare le informazioni sui giocatori, il loro ruolo, le partite effettuate e le squadre avversarie.

7. Si devono gestire i dati relativi alle attività di una pizzeria: è necessario registrare le informazioni sui tavoli, sulle ordinazioni e sui camerieri.

8. Uno stabilimento balneare utilizza un DBMS per rappresentare in un database sia le prenotazioni di ombrelloni e lettini (disposti su file numerate da 1 a 5, in cui la fila 1 è quella più vicina al mare, e su colonne numerate da 1 a 20), sia i turni del personale di assistenza ai bagnanti.

9. I dipendenti di un'azienda sono classificati secondo dei ruoli (che individuano la retribuzione annua lorda) e ciascun dipendente fa parte di un solo reparto. Ogni reparto è diretto da un direttore (prescelto tra i dipendenti). Ciascun dipendente può partecipare a uno o più progetti (eventualmente nessuno).

10. Si devono gestire i dati relativi a un ospedale: è necessario registrare le informazioni relative a pazienti, reparti, medici e ricoveri.

11. Il campionato di serie A è composto da 16 squadre di calcio e vengono effettuare due partite (andata e ritorno) tra le stesse squadre: si organizzi un database che permetta la gestione degli incontri).

12. La fase finale del campionato del mondo di calcio è organizzata in 8 gironi ciascuno composto da 4 squadre provenienti da 6 raggruppamenti: si organizzi un database che permetta la gestione dei sorteggi per la definizione dei gironi finali.

Elementi del modello E-R: le relazioni (o associazioni)

IN QUESTA LEZIONE IMPAREREMO...

- a conoscere il concetto di relazione
- a individuare le tipologie e la cardinalità delle relazioni
- a utilizzare i simboli del modello E-R per descrivere le relazioni tra le entità

MAPPA CONCETTUALE

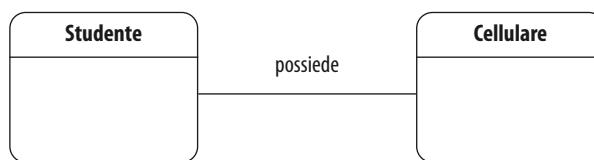
Relazioni (o associazioni)

Una relazione (*relationship*) rappresenta un'associazione tra una o più entità.

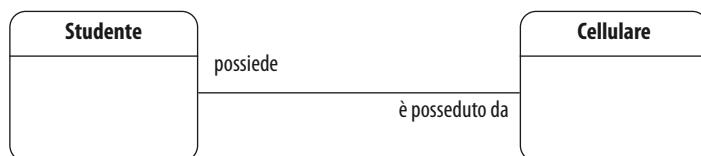
Una relazione (associazione) è spesso esprimibile attraverso un verbo: le frasi seguenti ne sono degli esempi.
– I docenti *insegnano* una materia.
– Gli alunni *frequentano* un corso di specializzazione.

– Gli impiegati *vengono assegnati* a un progetto.
– I dipartimenti *gestiscono* una o più attività.
– Una persona *guida* una automobile.
– I pazienti *vengono sottoposti* agli esami clinici.

Ogni associazione ha due versi con specifici significati e ogni verso si compone:
– di un'entità di partenza;
– di un'entità di arrivo;
– di una descrizione che consente di comprendere il significato dell'associazione.

Vediamo alcuni esempi.

ESEMPIO

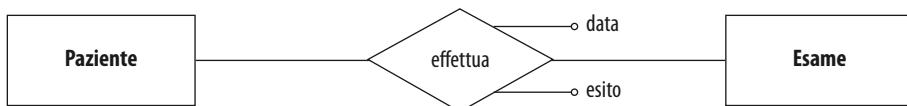
Tra l'entità Studente e l'entità Cellulare esiste un'associazione che può essere descritta nel linguaggio naturale secondo due prospettive: uno studente possiede uno (o più) cellulari e un Cellulare è posseduto da un solo Studente. Questa situazione si concretizza affermando che tra l'entità Studente e l'entità Cellulare esiste l'associazione Possiede.

Graficamente l'associazione viene rappresentata da una linea che unisce le due entità interessate con l'aggiunta di un rombo nel quale figura la descrizione dell'associazione.

Una grafica alternativa è quella che utilizza i diagrammi UML, come riportato di seguito.

A volte, per completezza, si indicano entrambe le forme dei verbi (che può essere attiva e passiva) relative alle due entità in relazione tra loro.

ESEMPIO

Se consideriamo la relazione "leggendola da destra verso sinistra", il verbo diventa "è posseduto", cioè abbiamo:

Le relazioni possono quindi essere lette in entrambe le direzioni, come vedremo in seguito:
– lo studente possiede il cellulare;
– il cellulare è posseduto dallo studente.

Attributi delle relazioni

Anche le relazioni, come le entità, possono avere degli attributi che le caratterizzano.

Esaminiamo un primo caso, dove individuiamo l'attributo della relazione tra Studente e Materia, cioè *"lo studente viene interrogato e **prende un voto** in una materia*; rappresentiamo la relazione con la notazione grafica "a rombo".

Prendiamo ora in considerazione un altro esempio, e individuiamo l'attributo della relazione tra Paziente ed Esame, cioè "*il paziente **oggi** effettua un esame con un certo **esito***".

In questo caso, si possono individuare più attributi che descrivono la relazione: vedremo in seguito come regolarci in queste situazioni nella lezione 8 dedicata alla semplificazione dei diagrammi E-R.

Anche nell'esempio seguente possiamo individuare più attributi che descrivono la relazione.

La notazione grafica classica è particolarmente comoda quando è necessario dettagliare la relazione, aggiungendo i rispettivi attributi, e non occorre indicare la direzione della relazione.

METTITI ALLA PROVA

➜ Attributi di una relazione

Descrivi gli attributi della relazione che esiste tra un utente di una biblioteca e un libro che viene preso in prestito.

Classificazione delle relazioni

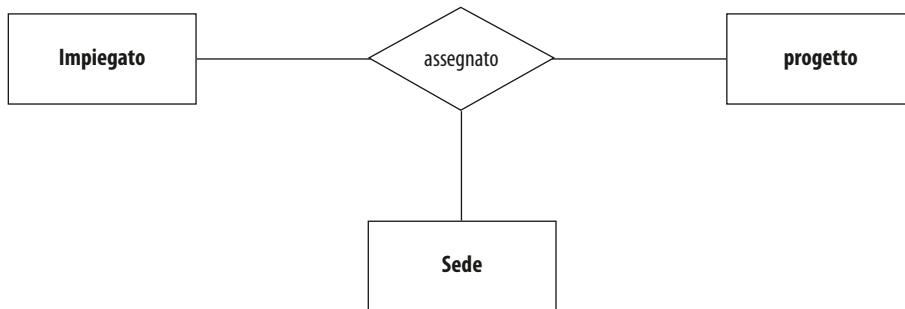
Le relazioni possono essere classificate in base a diversi criteri: noi le analizzeremo in base al grado e alla cardinalità.
Per completare il diagramma E-R è inoltre necessario aggiungere informazioni in merito alla direzione della relazione e alla eventuale presenza di situazioni gerarchiche tre le entità, che potrebbero comportare ambiguità e/o ridondanza nella successiva definizione dello schema logico.

Nei paragrafi che seguono esamineremo con attenzione cosa si intende con grado, cardinalità, direzione e gerarchia delle relazioni, in quanto un errore nella loro individuazione/classificazione in fase di progettazione concettuale può anche portare al fallimento dell'intero progetto.

Grado

Il grado di una relazione è il numero di entità associate alla relazione.

La relazione n-aria è la forma generale di grado n (enne). I casi speciali sono quelli binari e ternari, dove i gradi sono rispettivamente 2 e 3.

Le relazioni binarie, ovvero le associazioni tra due entità, sono le relazioni più comuni nel mondo reale e, di conseguenza, saranno quelle che verranno utilizzate nei nostri progetti.

ESEMPIO

Studente-Città

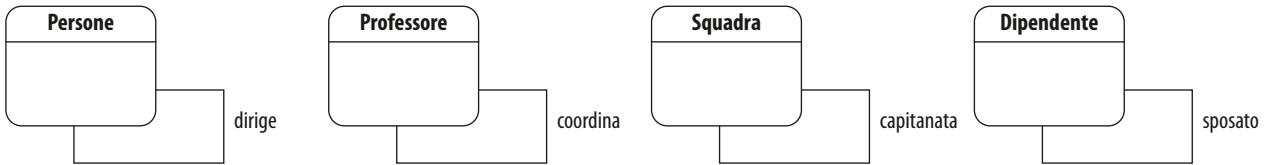
Supponiamo di avere le entità Studente e Città, che raccolgono le informazioni sugli studenti e sulla città dove abitano. Un esempio di relazione binaria è costituita da Residenza, che associa ogni Studente alla Città in cui risiede.

Una relazione ternaria implica tre entità e viene usata quando quella binaria è inadeguata: molti approcci di modellazione riconoscono solo relazioni binarie e in essi le relazioni ternarie o n-arie vengono decomposte in due o più relazioni binarie.

Un caso particolare di relazione è quella che “coinvolge una sola entità”: la relazione ricorsiva.

Una relazione è ricorsiva quando esiste una relazione tra un’entità e se stessa.

ESEMPIO

Consideriamo l’esempio dell’organizzazione di un ufficio: la relazione che lega impiegati con altri impiegati può essere indicata dal verbo “dirigere”, dove il “capufficio” dirige gli altri impiegati: *“un dipendente può dirigere altri dipendenti e ogni dipendente deve essere diretto da un solo dipendente”.*

Sono esempi di relazioni ricorsive:
– “il “coordinatore e il consiglio di classe”;
– “il capitano di una squadra sportiva”;

– "il capoclasse e gli alunni";
– "dipendenti sposati ad altri dipendenti".

Graficamente, la relazione ricorsiva viene rappresenta con il seguente schema:

Cardinalità

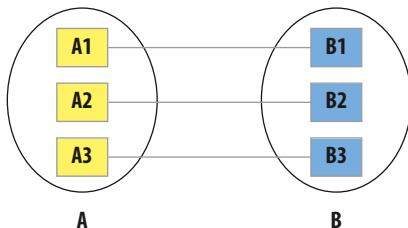
Per ottenere un modello adeguato del mondo reale, spesso è necessario classificare le relazioni a seconda del numero di entità associabili tra gli elementi delle coppie entità che stiamo mettendo in relazione.

Per ogni entità che partecipa a un'associazione bisogna indicare il numero minimo e massimo di istanze della relazione a cui un'istanza dell'entità può partecipare.
L'indicazione del numero delle istanze delle entità associate a una relazione prende nome di cardinalità.

Per indicare la cardinalità delle relazioni si ricorre a una coppia di numeri naturali. Quando la cardinalità non è nota con precisione, è anche possibile indicare con la costante n un numero generico maggiore di uno.

La cardinalità che si definisce su una relazione è un vincolo "di progetto" derivante dall'analisi della realtà che si vuole interpretare: si stabilisce una condizione che deve valere per tutte le istanze della entità a cui è associato il vincolo stesso.

Con riferimento alle cardinalità massime, abbiamo tre possibili situazioni:
– relazioni uno-a-uno: una relazione è uno-a-uno se a ogni istanza della prima entità corrisponde al più un'istanza della seconda entità e viceversa: viene anche indicata con (1, 1);
– relazioni uno-a-molti: una relazione si dice uno-a-molti se esiste un'istanza della prima entità cui corrisponde più di un'istanza della seconda ma a ogni istanza della seconda entità corrisponde al più un'istanza della prima entità: viene anche indicata con (1, n);
– relazioni molti-a-molti: una relazione si dice molti-a-molti se esiste un'istanza della prima entità in relazione con più di un'istanza della seconda, e viceversa: viene indicata con (n, n).

Rappresentazione UML

Nella rappresentazione UML sono possibili solo due valori di cardinalità, e cioè 1 oppure n, in modo da definire tre possibili combinazioni di relazioni:
– uno-a-uno (1, 1);
– uno-a-molti (1, n);
– molti-a-molti (n, n).

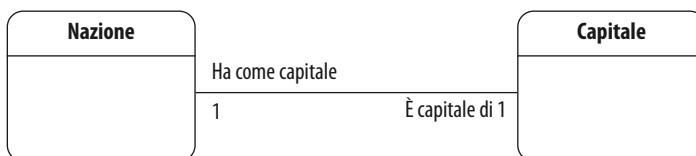
Di seguito le analizziamo dettagliatamente dato che sono la base per la costruzione del diagramma E-R.

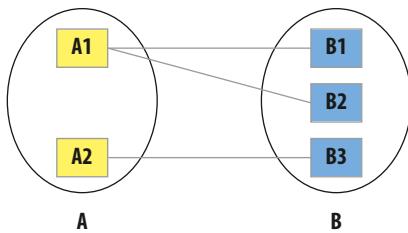
UNO A UNO (1, 1)

Date due entità A e B, la relazione uno-a-uno (1, 1) si ottiene quando al massimo una istanza dell'entità A viene associata a una sola istanza dell'entità B.

ESEMPIO

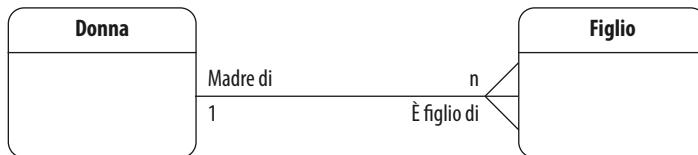
Un primo esempio di relazione uno-a-uno potrebbe essere il seguente.

Gli impiegati di una società dispongono di un proprio ufficio personale.

Per ogni impiegato esiste cioè un unico ufficio e per ogni ufficio esiste un unico impiegato.

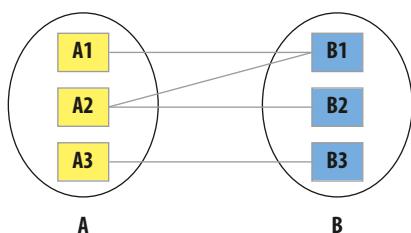
Vediamo ora un secondo esempio.

Ogni nazione ha una capitale.

Esiste cioè un'unica capitale per ogni nazione e una città può essere capitale di una sola nazione.

Graficamente, la relazione 1, 1 può essere così rappresentata.

UNO A MOLTI (1, n)

La relazione uno-a-molti (1, n) si ha quando, per un'istanza dell'entità A, ci sono zero, una, o molte istanze dell'entità B, ma per un'istanza dell'entità B c'è solo un'istanza dell'entità A.

ESEMPIO

Ecco alcuni esempi di associazioni uno-a-molti.

- Un ufficio *ha* molti impiegati, e ogni impiegato è *associato a un* ufficio.
- Una scuola *ha molti* alunni, e ogni alunno frequenta *una* scuola.
- In una città *abitano* molte persone, e ogni persona *abita* in una città.
- Uno studente *abita* in una città, in una città *abitano* diversi studenti.
- Una donna *ha più* figli, un figlio *ha una sola* madre.

Graficamente, l'associazione del lato n si rappresenta con un simbolo particolare che simboleggia una diramazione e che, per via della sua forma particolare, è indicato con il termine *crow's foot* (zampa di gallina).

MOLTI A MOLTI (n, n)

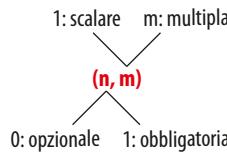
La relazione molti-a-molti (n, n), a volte chiamata non specifica, si ha quando per un'istanza dell'entità A ci sono zero, una o molte istanze dell'entità B e per un'istanza dell'entità B ci sono zero, una o molte istanze dell'entità A.

ESEMPIO

Ecco alcuni esempi di associazioni molti-a-molti.

- Un singolo impiegato può essere assegnato *a più progetti* e un singolo progetto può venire assegnato *a molti impiegati*.
- Un insegnante insegna *in più classi* e ogni classe ha *diversi insegnanti*.
- Un cantante *canta più canzoni*, una canzone può essere cantata da *diversi cantanti*.
- Un atleta partecipa *a più gare*, una gara è disputata *da più atleti*.

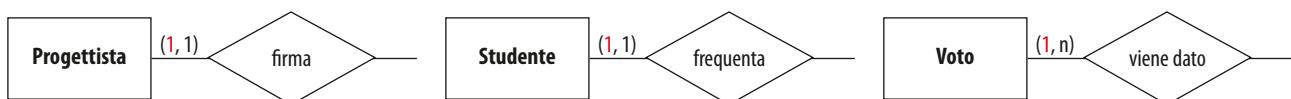
Graficamente, l'associazione presenta il simbolo di diramazione accanto a entrambe le entità.

Vedremo in seguito che le relazioni molti-a-molti non possono essere tradotte direttamente in tabelle relazionali: dovranno essere trasformate in due o più relazioni uno-a-molti usando delle entità associative nella fase di ristrutturazione del diagramma E-R.

Rappresentazione classica e vincoli di cardinalità

Per completezza, nei diagrammi E-R è sempre preferibile indicare, oltre ai tipi di cardinalità (1-n), anche i vincoli di cardinalità, che sono coppie di valori associati a ogni entità che partecipa a un'associazione in modo da specificare il numero minimo e massimo di istanze della relazione a cui un'istanza dell'entità può partecipare.

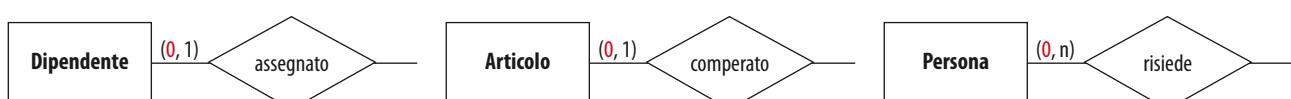
Utilizziamo la rappresentazione classica dei diagrammi E-R per indicare il vincolo di cardinalità aggiungendo una coppia di numeri tra parentesi (x, y) dove:
– x è un intero ≥ 0 ed è la cardinalità minima (min-card);
– y è un intero ≥ x ed è la cardinalità massima (max-card).

La cardinalità minima e massima viene indicata sul diagramma con (n, m) dove la prima cifra indica l'esistenza e la seconda la molteplicità.

L'esistenza (o minima cardinalità) di un'entità in una relazione indica che può essere obbligatoria oppure opzionale:
– 0 opzionale;
– 1 obbligatoria.

Se un'istanza di un'entità deve necessariamente esserci perché un'entità sia inclusa in una relazione, allora l'esistenza è obbligatoria.

ESEMPIO

Ecco alcuni esempi di esistenza obbligatoria.

- Un voto deve essere attribuito a uno studente (non ha senso un voto non attribuito).
- Uno studente deve frequentare un corso (non esistono studenti senza corsi di studio).
- Ogni progetto deve essere firmato da un tecnico (non possono esistere progetti senza il progettista).

Se l'istanza di un'entità non è richiesta, l'esistenza è opzionale.

ESEMPIO

Ecco alcuni esempi di esistenza opzionale.

- Un articolo a magazzino può essere ordinato da più clienti (ma possono esserci articoli che nessuno ha richiesto).
- Ogni persona può avere diversi recapiti (ma esistono anche persone che non hanno recapito).
- I dipendenti di una società possono essere assegnati a progetti (ma è possibile che un impiegato svolga mansioni slegate da qualsiasi progetto).

Vediamo un esempio dove figurano entrambe le situazioni, riportando l'esistenza sul diagramma.

ESEMPIO

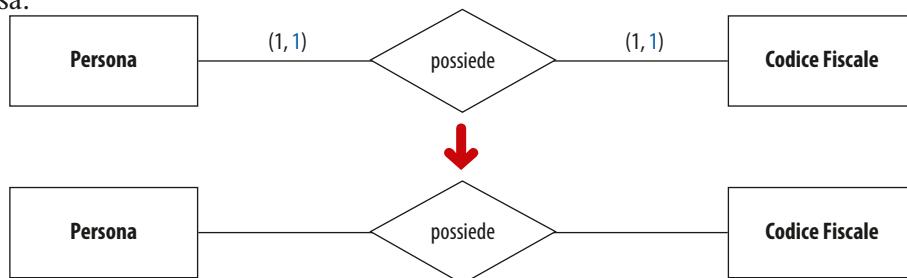
Ordine-Fattura

– a ogni ordine può essere associata nessuna (0) oppure una sola fattura;
– a ogni fattura è associato uno e un solo ordine (1).

La molteplicità (o massima cardinalità) indica il numero massimo di istanze che partecipano alla relazione:
– 1: una istanza;
– <un valore>: un numero massimo di istanze;
– n: senza limiti.

ESEMPIO

A ogni impiegato è assegnato sempre almeno un incarico e, al massimo, cinque incarichi (5), mentre un dato incarico può essere non assegnato a nessuno o, comunque, al massimo a cinquanta impiegati (50).
Il diagramma completo è il seguente.

Nel caso di relazione scalare (1) obbligatoria (1), generalmente il numero viene omesso sul diagramma, come nel seguente esempio.

Vediamo ora come rappresentare le tre situazioni classiche di una relazione binaria, cioè uno-a-uno (1, 1), uno-a-molti (1, n) e molti-a-molti (n, n) – illustrate nel precedente paragrafo con il diagramma UML – esprimendole in termini di molteplicità.

UNO A UNO (1, 1)

La relazione è uno-a-uno se la cardinalità massima di entrambe le entità è **1**.

Viene indicata come (1, 1) e generalmente, come prima anticipato, è sottointesa e, quindi, omessa.

UNO A MOLTI (1, n)

La relazione è uno-a-molti se la massima cardinalità verso una entità è **1** e la massima cardinalità verso l'altra entità è **n**: è indicata come (**1, n**).

MOLTI A MOLTI (n, n)

La relazione è molti-a-molti se la massima cardinalità verso entrambe le entità è **n**.

Vediamo ora un esempio completo discutendo ogni valore di esistenza.

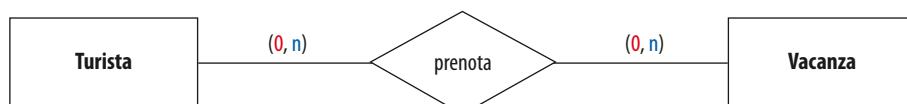
ESEMPIO

Persone-Automobili:

- minima cardinalità (Persona, proprietario) = 0: esistono persone che non posseggono alcuna automobile;
- massima cardinalità (Persona, proprietario) = n: ogni persona può essere proprietaria di un numero arbitrario di automobili;
- minima cardinalità (Automobile, proprietario) = 0: esistono automobili non possedute da alcuna persona;
- massima cardinalità (Automobile, proprietario) = 1: ogni automobile può avere al più un proprietario.

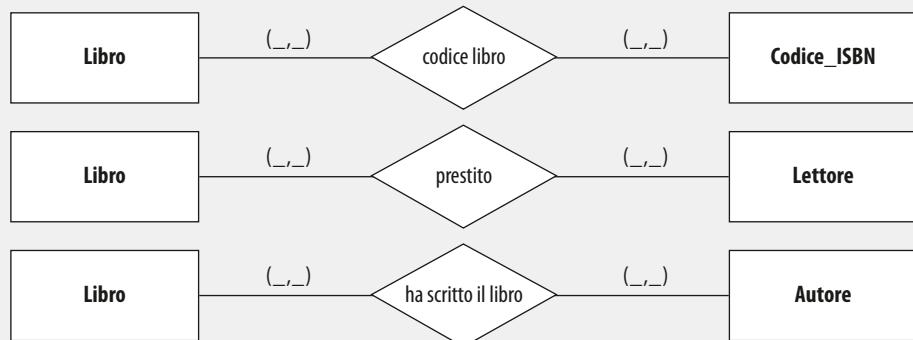
Quindi lo schema completo di vincoli è il seguente.

Riportiamo altri due esempi in modo da avere un caso per ogni diversa situazione.

ESEMPIO

Persona-Comune:

- a ogni Persona può essere associato un solo Comune di nascita;
- a ogni Comune possono essere associate da 0 a n nascite di Persone.

Turista-Vacanza:

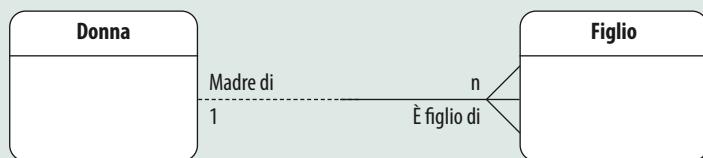
- ogni Turista può effettuare nessuna (0) oppure alcune (n) prenotazioni di una Vacanza.
- a ogni Vacanza possono essere associate nessuna (0) a tante (n) prenotazioni di Turisti.

METTITI ALLA PROVA

Vincoli di cardinalità

Aggiungi le cardinalità alle seguenti situazioni.

Entità	Cardinalità	Relazione	Cardinalità	Entità
Libro	(_,_)	codice libro	(_,_)	**Codice_ISBN**
Libro	(_,_)	prestito	(_,_)	**Lettore**
Libro	(_,_)	ha scritto il libro	(_,_)	**Autore**

PER SAPERNE DI PIÙ

L'OBBLIGATORIETÀ IN UML

Graficamente l'esistenza obbligatoria è rappresentata da una linea continua, mentre per l'esistenza opzionale si usa una linea tratteggiata. Per esempio, una persona può possedere (e guidare) diverse macchine, ma non tutte le persone hanno la patente e, quindi, guidano (o posseggono) un'automobile.

Rivediamo anche l'esempio delle madri e dei figli e, con un'analisi aggiuntiva, possiamo osservare che non tutte le donne hanno figli, quindi:

- ogni donna *può* essere madre di *uno* o più figli;
- ogni figlio *deve* avere *una sola* madre.

Introducendo l'opzionalità, il diagramma precedente diventa.

Dallo schema E-R deve essere possibile ricavare le frasi che ci hanno permesso la sua definizione: per fare ciò, ci vengono in aiuto le cosiddette "regole di lettura".

Direzione

È anche possibile completare il diagramma E-R introducendo un simbolo grafico che indica la direzione della relazione, nel caso che questa sia definita.

La direzione di una relazione indica l'entità da cui trae origine la relazione binaria: l'entità da cui si parte si chiama entità padre e l'entità a cui si arriva si chiama entità figlio.

La direzione di una relazione viene determinata dalla sua cardinalità:
- in relazioni di tipo uno-a-uno la direzione è dall'entità forte a quella debole; se entrambe sono forti, la direzione è arbitraria;
- nelle relazioni uno-a-molti, l'entità con cardinalità uno è il padre della relazione con cardinalità molti;
- la direzione nelle relazioni molti-a-molti è arbitraria.

ESEMPIO

Riconsiderando l'esempio delle entità Studente e Città, si vede che, poiché la relazione tra Città e Studente ha cardinalità uno-a-molti, la direzione è da Città a Studente.
L'entità Città è padre rispetto all'entità Studente.

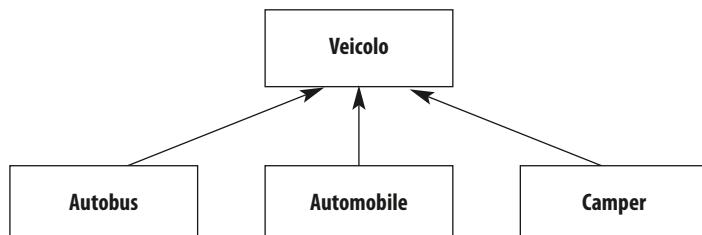
Si tratta di una notazione poco usata: abbiamo già visto nella lezione precedente una notazione grafica per indicare le chiavi esterne nelle entità deboli.

Anche nei diagrammi UML la direzione viene indicata con una freccia.

L'indicazione della direzione non riveste particolare importanza nei diagrammi E-R, in quanto le relazioni generalmente vengono in seguito rielaborate e ristrutturate, come vedremo nella prossima lezione, e quindi generalmente non viene indicata.

Relazione gerarchica o astrazione della generalizzazione

Esistono situazioni dove tra le entità può essere stabilita una gerarchia, come nella gerarchia delle classi della programmazione OOP (Object-Oriented Programming).
Osserviamo la seguente situazione.

- Beta è detta sottoclasse o specializzazione di Alfa.
- Alfa è detta superclasse o generalizzazione di Beta.

Nelle gerarchie sono presenti due vincoli (o proprietà).
- Vincolo di struttura: se Beta è sottoclasse di Alfa, Beta ha tutti gli attributi di Alfa e partecipa a tutte le associazioni cui partecipa Alfa (ereditarietà); questo non esclude che Beta possa avere altri attributi e partecipare ad altre associazioni.

– Vincolo di insieme: se Beta è specializzazione di Alfa, ogni oggetto di Beta è anche un oggetto di Alfa (cioè Beta è un sottoinsieme di Alfa).

Nella fase di progettazione dello schema E-R capita sovente di definire entità che hanno tra loro dipendenze gerarchiche. Vediamo un esempio.

ESEMPIO

Le istanze di Automobile sono un sottoinsieme delle istanze di Veicolo, ovvero, ogni automobile è un (is a) veicolo.
Ciò che caratterizza un veicolo caratterizza anche ogni suo sottoinsieme, ovvero ogni sottoclasse eredita dalla superclasse, ma può anche avere caratteristiche proprie.

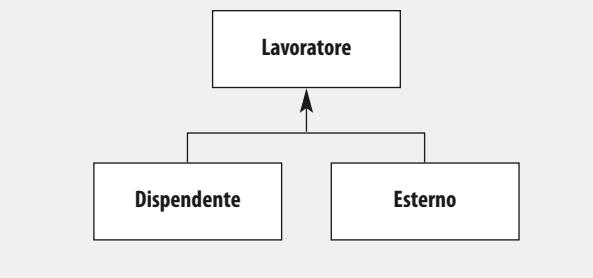
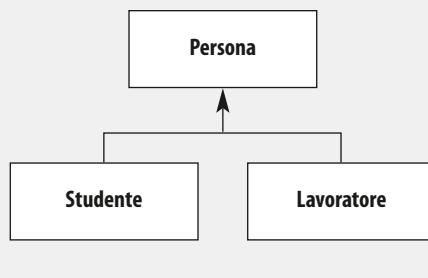
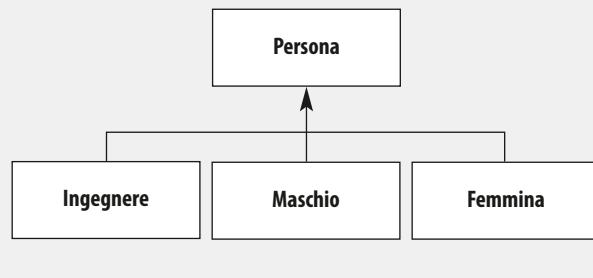
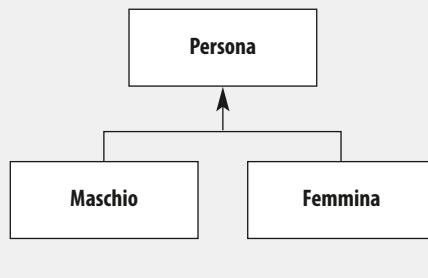
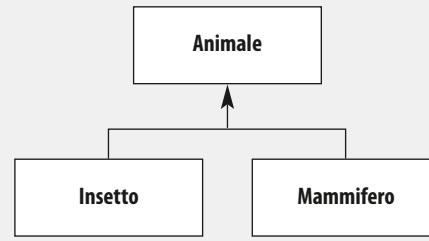
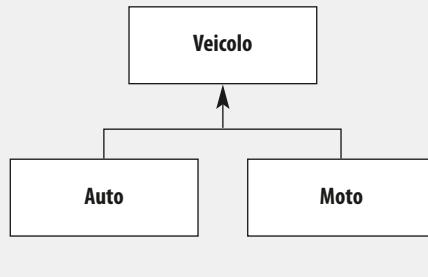
Le "sotto-entità" individuano un gruppo di elementi della classe padre, cioè specializzano e individuano un sottoinsieme di elementi: non è detto tuttavia che l'unione delle parti contempli tutta la casistica degli elementi della classe padre.
Nell'esempio precedente dei Veicoli non esiste infatti nessuna sotto-entità che individua i Trattori oppure i Camion, ecc.

Per indicare queste situazioni si introduce il concetto di copertura delle generalizzazioni, che può essere totale o parziale, esclusiva o sovrapposta.

Le generalizzazioni si caratterizzano quindi per "due dimensioni indipendenti":

1. confronto fra unione delle specializzazioni e classe generalizzata:
   – totale, se la classe generalizzata è l'unione delle specializzazioni;
   – parziale, se la classe generalizzata contiene l'unione delle specializzazioni;
2. confronto fra le classi specializzate:
   – esclusiva, se le specializzazioni sono fra loro disgiunte;
   – sovrapposta (overlapped), se può esistere una intersezione non vuota fra le specializzazioni.

Vediamo attraverso un esempio le quattro possibili combinazioni ottenibili da esse.

ESEMPIO

Le relazioni gerarchiche devono essere individuate durante la modellizzazione concettuale e trasformate (semplificate) durante la fase di ristrutturazione delle schema E-R per giungere alla definizione dello schema logico.

METTITI ALLA PROVA

→ Relazioni gerarchiche

Classifica le seguenti gerarchie.

Veicolo ← Auto, Moto

....

Animale ← Insetto, Mammifero

....

Persona ← Maschio, Femmina

....

Persona ← Ingegnere, Maschio, Femmina

....

Persona ← Studente, Lavoratore

....

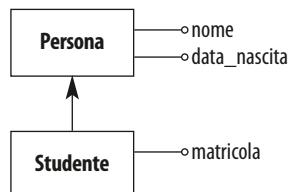
Lavoratore ← Dispendente, Esterno

....

Cittadino ← Cacciatore, Pescatore

....

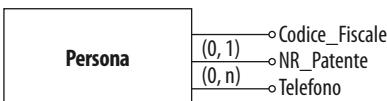
Laureato ← Legale, Ingegnere, Economista

....

Subset

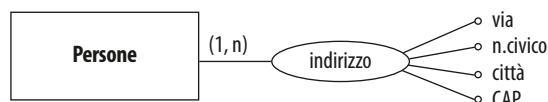
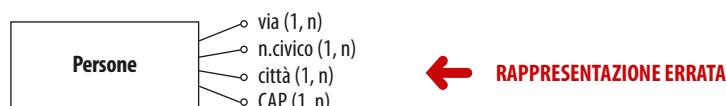
Un caso particolare di gerarchia è costituito dal subset, in cui si evidenzia una sola classe specializzata.

ESEMPIO

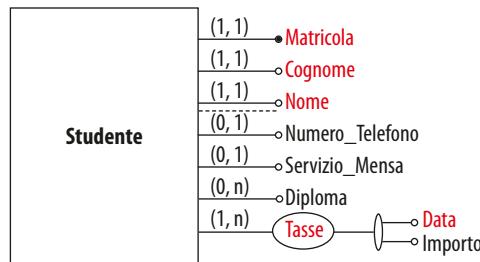
Nell'esempio che presentiamo, Studente eredita le proprietà di Persona e in più ha l'attributo matricola.

Nel caso del subset non ha senso parlare di tipo di copertura in quanto, essendo un sottoinsieme, il subset è incluso totalmente nell'entità dalla quale eredita parte degli attributi.

Cardinalità e obbligatorietà degli attributi

In fase di modellazione risulta comodo utilizzare la notazione usata per le entità, in merito alla molteplicità e alla obbligatorietà, anche per gli attributi.

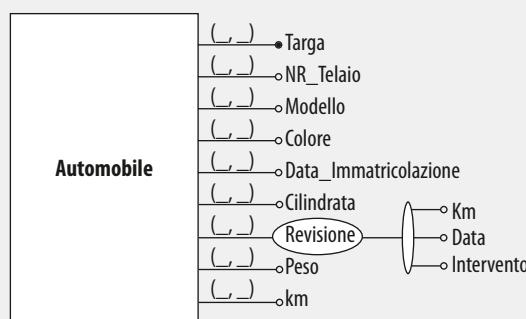
Abbiamo tre situazioni in cui l'attributo può essere:
- monovalore, se la cardinalità massima è 1, come per esempio il Codice_Fiscale;
- opzionale, se la cardinalità minima è 0, come per esempio il NR_Patente;
- multivalore (o ripetuti), se la cardinalità massima è n, come per esempio il Telefono.

Nel caso di presenza di più attributi multiplo (o multivalore), la creazione di un attributo composto può rendersi necessaria per evitare ambiguità.

ESEMPIO

Una persona può avere più indirizzi (residenza, abitazione, ufficio, ...) e questa situazione viene indicata con:

Sarebbe un errore replicare i singoli campi, anche perché si perderebbe la loro aggregazione.

Riportiamo a titolo riassuntivo un esempio, dove rappresentiamo l'entità Studente con i suoi attributi e le loro proprietà.

ESEMPIO

Gli attributi Matricola (Pk), Cognome e Nome sono obbligatori, mentre Numero_Telefono, Servizio_Mensa, Diploma (multiplo) sono opzionali; l'attributo Tasse è multiplo in quanto possono essere pagate più rate.

Nel seguito della trattazione utilizzeremo questa notazione per evidenziare alcune particolari situazioni complesse o di significativo interesse, mentre nei casi semplici utilizzeremo la notazione UML.

METTITI ALLA PROVA

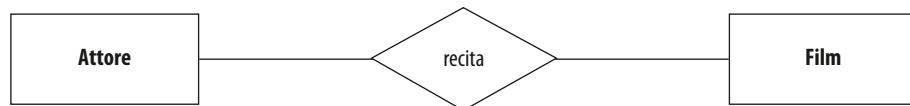
Cardinalità degli attributi • Obbligatorietà degli attributi

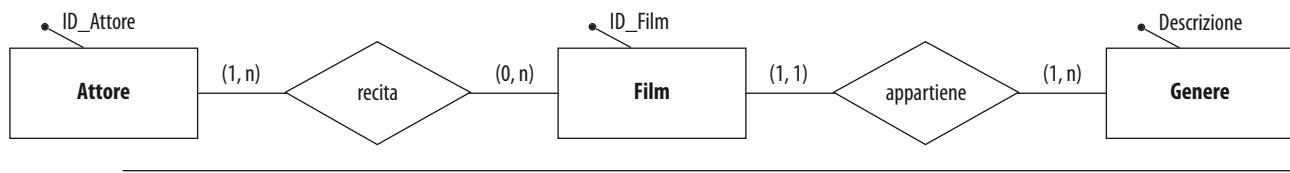
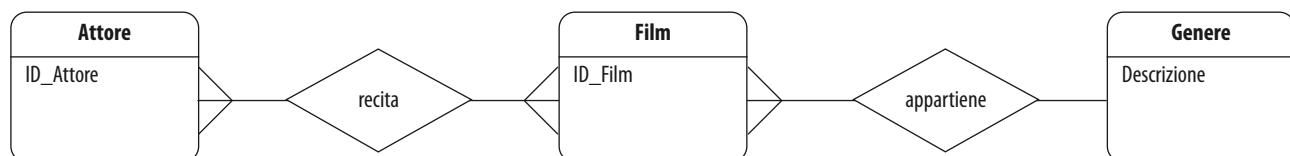
Nel diagramma della entità **Automobile** sono elencati gli attributi: classifica ogni attributo indicando obbligatorietà e molteplicità.

Esempio riepilogativo

Vediamo un esempio riassuntivo più complesso dove, dapprima, vengono analizzate separatamente le relazioni e, successivamente viene composto lo schema finale.

ESEMPIO

Si vogliono classificare i film di una videoteca casalinga per genere, evidenziando gli attori.
Il problema da risolvere può essere tradotto nelle seguenti frasi, che sintetizzano le due richieste del testo:

– un film viene classificato in un genere;
– in un film recitano alcuni attori e un attore può recitare in più film.

Per la prima richiesta il diagramma E-R è il seguente.

Per la seconda richiesta il diagramma E-R è il seguente.

Quindi, lo schema completo si ottiene "fondendo i due schemi" dove abbiamo aggiunto le chiavi primarie alle singole entità e la cardinalità:

- un attore recita in uno o più film;
- in un film possono recitare più attori (0 per esempio nel caso di cartoni animati).

Considerando le cardinalità massime, possiamo dire che tra Attore e Film la relazione è (n, n) e tra Genere e Film la relazione è (1, n).

Lo stesso schema, descritto con la rappresentazione UML, è il seguente:

Strumento CASE

Uno strumento CASE è un software che permette la progettazione mediante strumenti visuali e/o grafici sollevando il progettista dalla scrittura di parti di codice grazie alla generazione automatica di tutto/o parte di esso.

L'utilizzo diffuso della simbologia UML è anche dovuta al fatto che UML viene utilizzato dagli **strumenti CASE** disponibili per la progettazione assistita di database (come per esempio MySQL Workbench o DBDesigner), oppure nei programmi di disegno (come smartdraw, Visio e ArcGIS Diagrammer).

VERIFICA... le conoscenze

COMPLETAMENTO

Completa le frasi seguenti scegliendo tra le parole proposte.

1 Il modello Entità-Relazioni è assolutamente .... dal linguaggio scritto o parlato e permette quindi di comunicare agevolmente la struttura del database.
coerente – simile – svincolato – alternativo

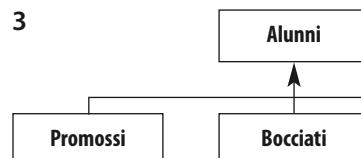
2 Ogni .... del modello E-R serve a rappresentare graficamente un concetto, concreto o astratto, del mondo reale che andiamo a modellare.
relazione – entità – attributo – chiave

3 Nella rappresentazione UML sono possibili solo due tipi valori di ...., e cioè 1 oppure n, in modo da definire tre possibili combinazioni di relazioni: uno a uno (1, 1), uno a molti (1, n) e molti a molti (n, n).
valore – simbolo – dato – elemento – cardinalità

4 L'esistenza di una entità in una relazione viene definita .... o ....
riflessiva – obbligatoria – simmetrica – opzionale – assoluta

5 Utilizziamo la rappresentazione classica dei diagrammi E-R per indicare il vincolo di cardinalità aggiungendo una coppia di .... tra parentesi: l'obbligatorietà della relazione viene indicata con il vincolo di cardinalità indicando il valore .... come valore minimo (cioè numero di sinistra nella parentesi), mentre l'opzionalità si indica con il valore ....
dati – numeri – lettere – simboli – 1 – 0 – N – X

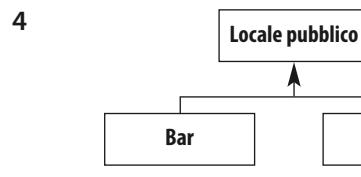
SCELTA MULTIPLA

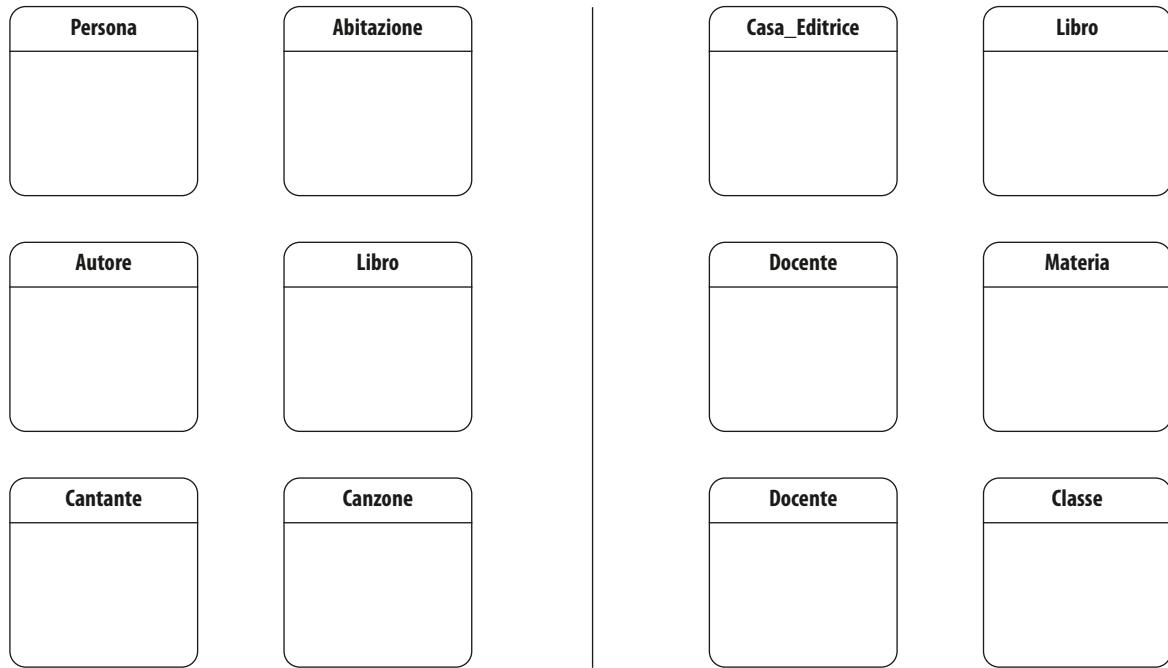
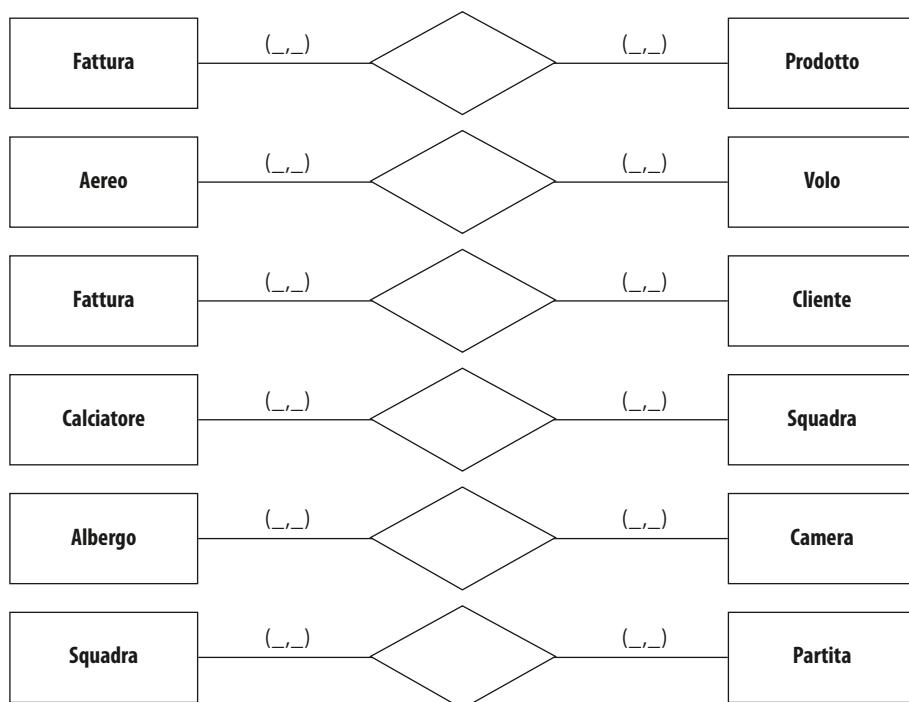
Individua il tipo di gerarchia

1

a totale-esclusiva
b parziale-sovrapposta
c totale-sovrapposto
d parziale-esclusiva

2

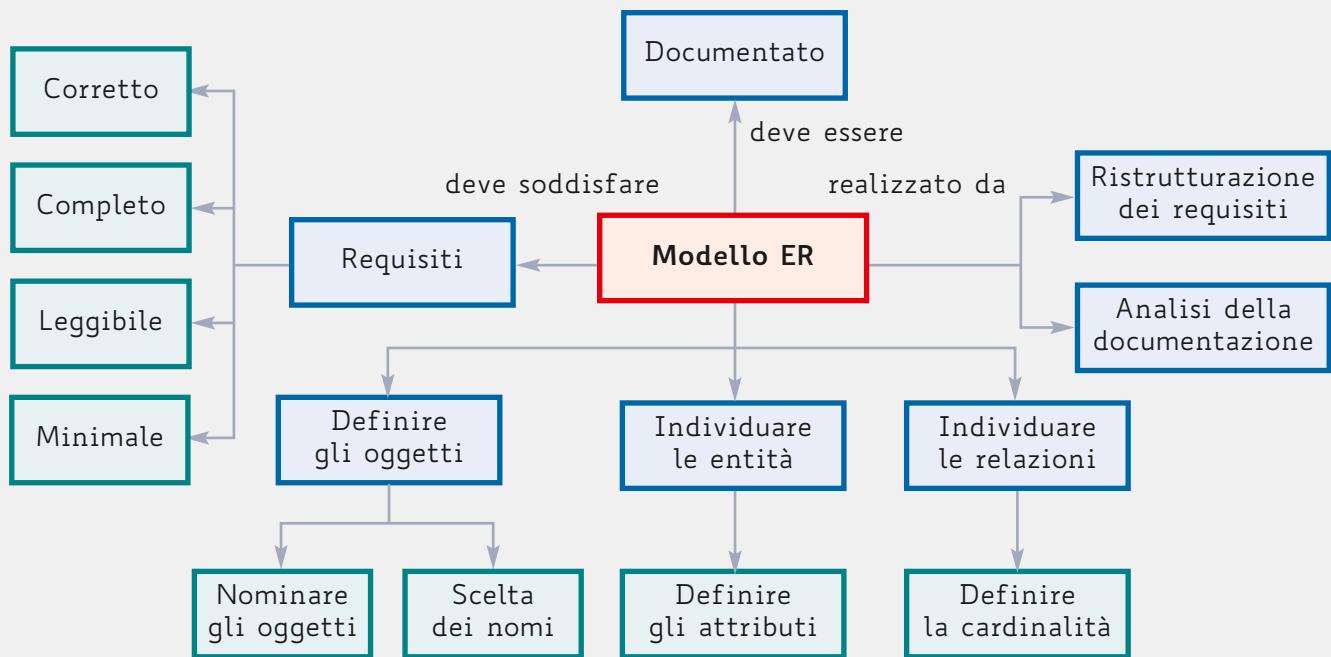
a totale-esclusiva
b parziale-sovrapposta
c totale-sovrapposto
d parziale-esclusiva

3

a totale-esclusiva
b parziale-sovrapposta
c totale-sovrapposto
d parziale-esclusiva

4

a totale-esclusiva
b parziale-sovrapposta
c totale-sovrapposto
d parziale-esclusiva

VERIFICA... le competenze

PROBLEMI

1 Completa i seguenti schemi E-R indicando gli attributi principali delle diverse entità e il tipo e la molteplicità della relazione

2 Completa gli schemi indicando il tipo, la minima e massima cardinalità della relazione e gli attributi principali di ciascuna entità

Definizione del modello E-R

IN QUESTA LEZIONE IMPAREREMO...

- a realizzare un modello E-R
- a perfezionare il modello E-R

MAPPA CONCETTUALE

didattica inclusiva

Modello ER — deve soddisfare → Requisiti → Corretto, Completo, Leggibile, Minimale

Modello ER — deve essere → Documentato

Modello ER — realizzato da → Ristrutturazione dei requisiti; Analisi della documentazione

Modello ER → Definire gli oggetti → Nominare gli oggetti; Scelta dei nomi

Modello ER → Individuare le entità → Definire gli attributi

Modello ER → Individuare le relazioni → Definire la cardinalità

Introduzione

In questa lezione verranno illustrati i concetti fondamentali per la realizzazione del modello E-R e per il successivo passaggio allo schema relazionale, elencando le operazioni normalmente compiute dal database designer, ovvero colui che ha il compito di definire, assieme all'utente del prodotto, il database.

Il database designer è responsabile dell'astrazione dei dati dal mondo reale a partire dall'analisi dei requisiti fino a ottenere la corretta modellazione degli stessi dapprima nello schema concettuale e successivamente nello schema logico: il suo compito è scomponibile in passi successivi che esamineremo nel dettaglio.

Il primo passo per costruire un database è quello di realizzare uno schema concettuale, che per noi consiste nella realizzazione del diagramma E-R, a partire dal quale si definisce il modello relazionale.

La realizzazione dello diagramma E-R è molto "delicata" in quanto commettere errori in questa fase può compromettere del tutto il "buon funzionamento" del database causando il fallimento del progetto.

Per giungere al diagramma E-R possiamo individuare due attività da compiere, tra loro sequenziali:
– realizzazione "provvisoria" del modello E-R (o "modello di base");
– raffinazione e ristrutturazione del modello E-R.

In questa lezione affrontiamo la realizzazione "provvisoria" del diagramma E-R, che prevede anch'essa alcune fasi specifiche che ne articolano lo sviluppo:
– per prima cosa si definiscono gli oggetti che andranno a comporre il diagramma;
– successivamente si completano le entità con gli attributi;
– infine si procede con l'individuazione delle relazioni esistenti tra le entità.

Nella prossima lezione effettueremo invece le operazioni di ristrutturazione per ottenere il diagramma E-R finale dal quale derivare lo schema logico.

Individuazione degli oggetti del diagramma

Il primo compito di un database designer consiste nell'analizzare le informazioni raccolte durante l'analisi dei requisiti al fine di costruire il modello di base, che andrà poi raffinato e ristrutturato fino al suo completamento.

Le prime operazioni che il modellatore esegue hanno il compito di classificare gli oggetti come entità oppure attributi partendo dalla documentazione del progetto.

Questa fase viene realizzata mediante due operazioni:
– raccolta e analisi della documentazione;
– definizione del glossario dei termini.

Analisi della documentazione

Durante l'analisi dei requisiti viene prodotta e analizzata una serie di documenti, che può essere classificata per tipologia:
– documentazione specifica prodotta per il progetto, che comprende:
  • le note delle riunioni tecniche e le richieste del cliente;

- gli appunti sulle interviste agli utenti finali;
  - la documentazione scritta predisposta appositamente;
- documentazione esistente, composta da:
  - le normative generali e del settore;
  - i regolamenti interni;
  - le procedure aziendali;
- sistema esistente, in particolare:
  - la documentazione del sistema da rimpiazzare;
  - le specifiche di integrazione con sistemi esistenti.

Il progettista si deve basare su tutta questa documentazione non solo nella prima fase ma durante tutto il progetto del database: deve produrre una prima versione dei requisiti in linguaggio naturale, raggruppando frasi relative a categorie diverse di dati e di operazioni.

Di ogni testo deve analizzare le frasi per eliminare ambiguità provocate da:
- pluralismo di percezione;
- incompletezze di descrizione;
- omonimie;
- sinonimie;
- conflitti di descrizione;
- similitudini.

ESEMPIO

Riportiamo a titolo di esempio alcune situazioni che generano ambiguità:
- sinonimi come "docente", "insegnante", "istruttore" vanno uniformati e ricondotti a un unico termine;
- casi di omonimia come "*posto* di lavoro" e "*posto* di nascita" possono generare problemi: meglio sostituire la prima espressione con "impiego" e la seconda con "città natale"; analogamente, "*luogo* dove si tengono le lezioni" e "*luogo* dove si è nati" sono espressioni che è meglio sostituire rispettivamente con "aula" e con "città natale".

Glossario dei termini

Per ogni singola parte del progetto o sottosistema che si sta realizzando si devono ricontrollare insieme ai responsabili di settore le frasi relative alle varie categorie di dati e alle operazioni che li coinvolgono, cercando di eliminare le ambiguità del linguaggio naturale, e si deve costruire, a partire dalle frasi, un glossario di termini che, per esempio, contenga per ogni termine la descrizione, l'elenco dei sinonimi e l'elenco dei termini a cui si collega.

Il glossario fa parte della documentazione del progetto.

Su di esso e/o mediante esso si eseguono due controlli:
- si verifica la completezza del glossario, cioè se tutti gli aspetti importanti sono stati considerati;
- si verifica la consistenza delle specifiche, se cioè tutti i termini sono stati definiti.

ESEMPIO

Supponiamo di dover analizzare il funzionamento di un magazzino: alcune frasi sintetiche che ne forniscono una descrizione sono le seguenti:
- *il magazzino è composto da scaffali*
- *gli scaffali contengono prodotti*
- *i fornitori forniscono prodotti*
- *gli operai sono addetti agli scaffali*
- *i clienti ordinano prodotti*

Se ne può ricavare il seguente glossario, che riportiamo in tabella.

TERMINE	DESCRIZIONE	SINONIMO	LEGAME
Fornitore	Partita IVA Denominazione Indirizzo Nr. telefonico ecc.		Prodotto
Cliente	Partita IVA Denominazione Indirizzo Nr. telefonico ecc.	Acquirente	Prodotto
Prodotto	Codice Nome Genere …	Articolo Voce	Fornitore Cliente Scaffale
Scaffale	Supporto numerico	Ripiano Armadio	Operaio Prodotto
Operaio	Dati anagrafici Matricola Qualifica	Addetto Magazziniere	Scaffale

Spesso, la compilazione del glossario viene trascurata (soprattutto da parte degli alunni inesperti) in quanto viene reputato una "perdita di tempo".
È invece buona abitudine utilizzare tutti gli strumenti di progettazione, anche quelli che sembrano "più inutili", perché il rispetto di una corretta e completa metodologia sta alla base del raggiungimento di risultati efficaci di progettazione.

L'importanza del glossario viene "riconosciuta" quando si affrontano problemi in un ambito non conosciuto dal progettista, dove la documentazione raccolta dalle interviste "è piena" di termini specifici della disciplina per la quale viene richiesto lo sviluppo di una applicazione.

Definizione delle entità e degli attributi

Dopo aver esaminato la documentazione e costruito il glossario, il passaggio successivo consiste nell'individuare le entità distinguendole dagli attributi: è possibile seguire questo schema operativo:
– riorganizzare le frasi e definire gli oggetti;
– nominare gli oggetti;
– documentare il progetto: realizzare la matrice tra entità e attributi.

Il modello E-R non spiega come identificare gli oggetti che lo compongono, ma solo come definirli.

Esistono tuttavia delle linee guida che vengono di fatto seguite e che ci aiutano a identificare i tre oggetti che compongono i diagrammi, cioè le entità, gli attributi e le relazioni, in modo corretto.

Un concetto verrà rappresentato come:
– entità, se ha proprietà significative e descrive oggetti con esistenza autonoma;
– attributo, se è semplice, non ha proprietà e serve solamente per descrivere le entità;
– relazione (associazione), se correla due o più concetti/entità.

Descriviamo ora nel dettaglio ogni singola operazione prevista dal nostro schema operativo.

Riorganizzare le frasi e definire gli oggetti

Dopo aver realizzato il glossario ed eliminato le omonimie è utile riorganizzare le frasi raggruppandole in base al concetto cui si riferiscono.

Nell'esempio del magazzino del paragrafo precedente potremmo individuare e isolare le:
– frasi di carattere generale;
– frasi riferite ai fornitori, ai prodotti, ai clienti, agli operai.

Ogni gruppo di frasi viene analizzato assieme per definire le entità e gli attributi.
Ma "che cosa rende un oggetto un attributo piuttosto che un'entità"?

Non esiste una risposta definitiva a questa domanda: il lavoro del database designer, infatti, consiste nel modellare qualcosa di esistente nel mondo reale secondo uno specifico punto di vista dettato dai requisiti peculiari della singola situazione e, pertanto, la risposta alla domanda dipende dal progetto a cui si lavora.

ESEMPIO

Analizziamo un modello che descrive i libri presenti in una biblioteca; possiamo vedere che gli attributi e le entità assumono diversa collocazione in funzione del contesto che viene descritto.
– Se affrontiamo il problema dal punto di vista dello schedario abbiamo un'unica entità: Scheda_Biblioteca con attributi Autori, Titolo, Editore, Luogo, Edizione, Anno...
– Se invece lo osserviamo dal punto di vista dell'autore o dell'editore si hanno tre entità:
  • Autore con attributi Nome, Nazionalità, Anno, Nascita...;
  • Editore con attributi Nome, Indirizzo, Città...;
  • Libro con attributi Codice_ISBN, Collocazione...

Entrambi i modelli possono essere corretti e quindi non esiste un'unica schematizzazione, ma ogni situazione richiede un modello adeguato e personalizzato, anche se gli attributi e le entità possono essere simili o coincidenti.

Dopo questa premessa, proviamo a definire e descrivere entità e attributi.

Individuare le entità

Le entità sono i primi elementi da determinare nella progettazione concettuale di un database ed è possibile individuarle aiutandosi con alcune definizioni e osservazioni.

Viene considerata un'entità:
– qualunque oggetto per il quale è necessario salvare alcuni attributi;
– qualsiasi oggetto che deve essere rappresentato in un database.

In particolare, sono entità:
– qualsiasi persona, luogo, cosa, evento o concetto distinguibile, sul conto del quale le informazioni vanno memorizzate;
– qualsiasi cosa per la quale salviamo informazioni (per esempio fornitori, macchine, impiegati, numero posti in aereo ecc.).

Non tutto ciò su cui l'utente vuole raccogliere informazioni diviene un'entità: un concetto complesso potrebbe richiedere più di un'entità per essere rappresentato, mentre altre "cose" che l'utente ritiene importanti potrebbero non essere entità.

Definire gli attributi

Gli attributi, come abbiamo già detto, sono gli oggetti che descrivono un'entità: corrispondono ai campi dei record.

Nell'individuazione degli attributi è necessario avere presenti alcune regole fondamentali.

1. Gli attributi devono essere atomici, ovvero presentare un singolo fatto o una singola informazione, dove per singola informazione si intende proprio la più piccola informazione salvabile.

ESEMPIO

Nel caso in cui si voglia salvare il nome di una persona, risulta conveniente salvare in attributi diversi il nome e il cognome. La necessità di utilizzare dati atomici, come vedremo, è anche richiesta nel processo di normalizzazione dei dati, operazione che svolgeremo in seguito.

Non utilizzare i dati in forma atomica porta alla generazione di un insieme di errori: illustriamo di seguito i più frequenti.

- Aggregazioni semplici: l'esempio precedente, che concatena il nome e il cognome, è un caso di aggregazione semplice. Un altro errore da aggregazione semplice viene fatto sull'attributo Indirizzo, dove spesso si concatena la via, la città e il codice postale. È bene invece separare questi dati anche quando l'utente finale li usa solo assieme, per aumentarne la flessibilità di utilizzo (si possono ordinare i dati per CAP).
- Codici complessi: alcuni attributi hanno per valore codici composti di parti di informazioni concatenate che nella vita quotidiana sono utilizzate assieme, ma che risulta opportuno scomporre nei database. Un esempio di codice complesso è il numero telefonico completo di prefisso, che deve essere spezzato in due attributi.
- Attributi testuali: ci sono attributi che vengono definiti erroneamente di tipo testo perché questa modalità di rappresentazione risulta essere la più semplice.
  A volte questo errore non genera problemi immediati, soprattutto se su questi attributi non sono previste elaborazioni; esigenze successive potrebbero tuttavia richiedere operazioni non previste su questi campi che, essendo in formato testo, genererebbero sicuramente gravi problemi (richiedendo, per esempio, la completa conversione "di tipo" su tali campi, operazione a volte di difficile realizzazione). Un esempio classico è quello di definire le date (GG-MM-AAAA) con attributi di tipo testo, e dover poi eseguire su di esse delle operazioni.
- Attributi numerici: ci sono attributi che vengono definiti erroneamente di tipo numerico perché questa modalità di rappresentazione "sembra essere" la più indicata. Un esempio classico è quello di definire di tipo intero il CAP, la partita IVA, il numero civico, il numero di telefono ecc., quando invece andrebbero definiti di tipo testo, in quanto su di essi non verranno mai effettuate operazioni matematiche!

2. Gli attributi derivati non dovrebbero essere memorizzati.
Gli attributi derivati sono quelli ottenuti come risultati dall'applicazione di una formula o da operazioni di elaborazione su altri attributi. Le argomentazioni contro l'inclusione dei dati derivati sono basate sulla premessa che, in quanto derivati, non c'è alcun bisogno di salvare questi attributi nel database dato che possono sempre essere ricavati nuovamente.

Su questo punto gli esperti e i teorici, che dovrebbero indicare delle regole di base, a volte sono in disaccordo tra di loro: a volte la "comodità" di avere un dato già pronto senza doverlo ricalcolare prevale sulla buona regola di non inserire un attributo derivabile (e quindi ridondante).

3. Utilizzare codici per classificare gli attributi ogni volta che si presenta tale opportunità.
Un valore codificato usa un numero ridotto di lettere o numeri per rappresentare un dato: per esempio, i dati da memorizzare per il campo Sesso potrebbero essere semplicemente le lettere M e F , piuttosto che la scritta “Maschio” o “Femmina”.
Come vedremo, la codifica porta notevoli vantaggi nella gestione delle informazioni, che vanno tuttavia a scapito della semplicità di interpretazione e leggibilità da parte dell’utente.

Anche in questo caso, i teorici si dividono in due opposte fazioni:
- alcuni sono contro questa pratica sostenendo che i codici non hanno significato intuitivo per l’utente finale e aggiungono complessità all’elaborazione dei dati;
- altri, invece, sono favorevoli ricordando che molte organizzazioni hanno una lunga storia di uso di attributi codificati, e che inoltre i codici salvano spazio e aumentano la flessibilità, poiché possono facilmente essere aggiunti o modificati mediante tabelle di transcodifica.

Queste tre regole devono sempre essere adottate, salvo casi particolari nei quali potrebbero esserci situazioni tali per cui la loro osservanza potrebbe aumentare la complessità di realizzazione del database.

Inoltre, è anche importante classificare gli attributi considerando se la situazione fotografata è:
- permanente, cioè vale per ogni giorno e mese dell’anno;
- temporanea, cioè se i dati variano in funzione del tempo, come i turni settimanali, le sospensioni estive, le fatture dell’anno in corso, i giustificativi per la dichiarazione dei redditi ecc.

Queste osservazioni portano, nella fase successiva di raffinamento dello schema E-R, alla definizione di nuove entità, come quelle di storicizzazione o off-line, nelle quali sono presenti dati “del passato” ma sempre indispensabili per soddisfare tutte esigenze per il completo utilizzo del database.

Nominare gli oggetti

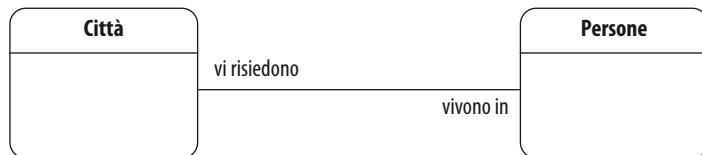
Per non correre il rischio di trovarsi in presenza di omonimi, le definizioni devono essere più complete e accurate possibile affinché tutte le parti che concorrono alla modellazione dei dati sappiano quali concetti gli oggetti vogliono rappresentare.

La condizione ideale si ha quando l’utente finale fornisce le definizioni assieme ai requisiti del sistema che dobbiamo creare durante la fase di analisi e di definizione delle specifiche: essendo una condizione ideale, però, non viene quasi mai raggiunta!

Come abbiamo accennato nel paragrafo precedente, nella fase di definizione degli oggetti il modellatore dovrebbe fare attenzione nel risolvere qualsiasi istanza individuando i casi nei quali una singola entità rappresenti effettivamente due concetti diversi (omonimia) o dove due diverse entità rappresentano effettivamente la stessa cosa (sinonimia).

Questa situazione tipicamente si verifica quando vengono intervistate più persone con compiti e livelli operativi diversi all’interno della stessa organizzazione, perché ciascuno ha conoscenza solo del proprio “ambito di lavoro” e pensa a eventi o processi secondo il “proprio personale punto di vista”.

Scelta dei nomi

Tutti gli oggetti che fanno parte del nostro modello “dovrebbero” avere un nome e anche questo aspetto deve essere regolato da una serie di proprietà.

I nomi degli oggetti devono:
- essere unici;
- avere un significato per l'utente finale;
- contenere un numero minimo di parole di cui si ha bisogno per descrivere univocamente e accuratamente l'oggetto.

Per le entità e gli attributi i nomi sono usati generalmente al singolare, mentre le relazioni sono tipicamente descritte attraverso verbi.

ESEMPIO

Una semplice relazione tra Città e Persone può essere così rappresentata in UML.

oppure con un unico termine per la relazione nel diagramma E-R.

È sconsigliato usare abbreviazioni o acronimi perché possono ingenerare confusione circa il loro significato, tranne per quelli "universalmente riconosciuti e utilizzati" come il CAP, l'IVA, l'ISBN, la TASI, la TARI, l'indirizzo IP ecc.
Il rischio, in ogni caso, è di utilizzare nomi uguali per rappresentare concetti diversi: in questo caso, ci si troverebbe di fronte a degli omonimi che renderebbero inutilizzabile il modello date che, come ben sappiamo, in informatica "non si può essere ambigui"!

Documentare il progetto: matrici tra entità e attributi

Oltre al glossario, per tenere traccia degli oggetti che faranno parte del database si possono usare due schemi: la matrice entità-entità e la matrice entità-attributo.

Nel dettaglio:
- la matrice entità-entità è un tabella bidimensionale per indicare le relazioni tra le entità. I nomi di tutte le entità identificate sono elencate sui due assi (verticale e orizzontale). Una relazione viene indicata con una "X" posizionata nelle caselle di intersezione tra due entità. Questa tabella sarà il punto di partenza per la fase di progettazione concettuale dove ogni relazione deve essere descritta mediante uno schema E-R;
- la matrice entità-attributo viene usata per indicare l'assegnazione degli attributi alle entità. È simile nella forma alla matrice entità-entità, tranne che i nomi degli attributi sono elencati in righe: questa tabella viene utilizzata successivamente, in fase di definizione del modello logico, dove per ogni tabella vengono indicati i singoli campi (tracciato record).

Queste due tabelle, assieme al glossario, fanno parte della documentazione del progetto e dovrebbero essere contenute in appositi documenti utilizzabili in futuro, in occasione di integrazioni e/o manutenzioni del sistema.
Non esistono standard per l'organizzazione di questi documenti, e generalmente vengono realizzati direttamente "su carta" o utilizzando un programma di videoscrittura.

ESEMPIO

Si vogliono definire le associazioni e gli attributi per organizzare i dati relativi ai dipendenti di una società allo scopo di gestire le informazioni concernenti i progetti seguiti dagli impiegati, le attività da svolgere all'interno di un progetto e le conoscenze necessarie per i progetti.

Matrice entità-entità

	IMPIEGATO	PROGETTO	ATTIVITÀ	CONOSCENZE
Impiegato		X		
Progetto	X		X	X
Attività		X		
Conoscenze	X			

Matrice entità-attributo

	IMPIEGATO	PROGETTO	ATTIVITÀ	CONOSCENZE
Matricola impiegato	X			
Nome impiegato	X			
Mansione				
Codice progetto		X		
Nome progetto		X	X	
Attività			X	
Descrizione attività			X	
Conoscenza				X
Specializzazione				X

Spesso l'importanza di queste due semplici tabelle viene sottovalutata, così come l'utilizzo del glossario, e la progettazione viene effettuata direttamente con la stesura dei diagrammi E-R, che a volte comporta la realizzazione di modelli errati e confusi.
Come già detto in precedenza, è meglio utilizzare sempre tutti gli strumenti che si hanno a disposizione per realizzare un progetto in modo da limitare le possibilità di errori.

Individuazione delle relazioni

Come abbiamo già detto, per individuare le relazioni si analizzano i verbi che figurano nelle frasi scritte in fase di analisi, dato che ogni relazione è indicata generalmente da un verbo che permette di connettere due o più entità.

ESEMPIO

Nella frase "*gli studenti* ***ricevono*** *voti*" l'entità studente è messa in relazione con l'entità voto attraverso il verbo "ricevono".
Nella frase "*una persona* ***possiede*** *uno o più telefoni*" l'entità persona è messa in relazione con l'entità telefono attraverso il verbo "possiede".

Tra tutte le entità individuate nella prima fase di progetto è necessario individuare almeno una relazione che le lega, altrimenti risultano isolate.

A partire dalla matrice entità-entità si deve:
– definire e individuare tutte le relazioni;

– classificare le relazioni in termini di:
  - cardinalità, per quantificare le relazioni tra le entità misurando come molte istanze di una entità sono relazionate a singole istanze di un'altra entità;
  - opzionalità, per verificare se la relazione deve esistere oppure è opzionale;
  - direzione, per individuare la direzione della relazione (per esempio da studente a voto);
  - dipendenza, per stabilire se una relazione dipende da una o più altre relazioni.

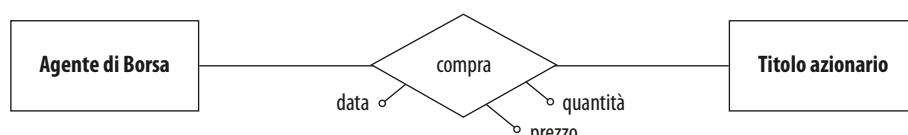
Una volta individuate le entità, si definiscono le possibili gerarchie in termini di:
– generalizzazione, se un caso è più generale di un altro;
– specializzazione, se un caso è più particolare di un altro.

Attributi delle relazioni

La rappresentazione grafica dello schema ci permette di poter aggiungere gli eventuali attributi nel caso che questi siano legati specificamente alla relazione e non alle entità.

Vediamo un primo esempio.

ESEMPIO

Descriviamo la relazione che deriva dalla frase "*lo studente è **interrogato** in una materia*".
Se analizziamo la relazione, che in questo circostanza è espressa da interrogazione, ci rendiamo immediatamente conto che per specificare l'interrogazione è necessario indicare in quale data si è svolta e il voto che è stato attribuito: sono due attributi legati proprio alla relazione, in quanto non appartengono né allo Studente né alla Materia.

Vediamo adesso un secondo esempio.

ESEMPIO

Descriviamo la relazione che deriva dalla frase "*un agente di borsa **compra** un titolo azionario*".
Per descrivere la relazione espressa da compra è necessario indicare come attributi la data dell'operazione, la quantità e il prezzo del titolo acquistato.

Conclusione

Lo schema E-R deve essere verificato accuratamente affinché risponda a requisiti di:
– correttezza: non devono essere presenti errori (sintattici o semantici);
– completezza: tutti i dati di interesse sono specificati;
– leggibilità: anche in relazione ad aspetti prettamente estetici dello schema;
– minimalità: è importante capire se esistono elementi ridondanti nello schema e se queste situazioni costituiscono un problema oppure sono dovute a una scelta di progettazione volta a favorire l'esecuzione di certe operazioni.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Nella realizzazione del diagramma E-R provvisorio, nella prima fase:
a si individuano le entità;
b si definiscono gli oggetti
c si classificano gli attributi
d si individuano le relazioni

2 Nel glossario si verificano:
a la completezza e la consistenza
b la completezza e l'unicità
c l'unicità e la consistenza
d i sinonimi e gli omonimi

3 Negli attributi è possibile utilizzare:
a aggregazioni semplici
b codici complessi
c attributi testuali
d attributi derivati

4 Nella documentazione del progetto si usa: *(2 risposte)*
a la matrice entità-attributo
b la matrice entità-entità
c la matrice entità-relazione
d la matrice attributo-attributo

5 A partire dalla matrice entità-entità si deve definire:
a cardinalità
b opzionalità
c direzione
d dipendenza
e obbligatorietà

6 Le relazioni possono essere: *(indicare le risposta errata)*
a ricorsive
b iterative
c 1, 1
d 1, n
e n, n
f riflessive

VERO/FALSO

1 La realizzazione del diagramma E-R ha come prima fase l'individuazione delle entità e la definizione degli attributi
2 Per le entità possiamo esprimere le gerarchie come generalizzazione o specializzazione. V F
3 Il modellatore deve risolvere i casi nei quali una singola entità rappresenti effettivamente due concetti diversi (omonimia). V F
4 Il modellatore deve risolvere i casi nei quali due diverse entità rappresentano la stessa cosa (sinonimia). V F
5 Il glossario di termini indica l'elenco dei sinonimi. V F
6 Gli attributi identificano o descrivono le entità. V F
7 Le relazioni sono associazioni fra attributi. V F
8 Gli attributi devono essere atomici ovvero presentare un singolo fatto. V F
9 La matrice entità-entità è una tabella che indica le relazioni tra le entità omogenee. V F
10 A partire dalla matrice entità-entità si devono definire, individuare e classificare tutte le relazioni. V F
11 Lo schema E-R deve rispondere al requisito di minimalità. V F

Verifica di fine Lezione

VERIFICA... le competenze

AREA DIGITALE

Esercizi per il recupero e il rinforzo

PROBLEMI

1 Disegna il diagramma E-R per gestire un archivio con informazioni su Cantanti e Dischi.
2 Disegna il diagramma E-R per gestire un archivio con informazioni su Immobili e rispettivi Proprietari.
3 Disegna il diagramma E-R per gestire un archivio con informazioni su Attori, Registi e Film.
4 Disegna il diagramma E-R per gestire un archivio con informazioni su Album a Fumetti e relativi Personaggi.
5 Disegna il diagramma E-R per gestire un archivio con informazioni su Libri, Autori e Case Editrici.
6 Disegna il diagramma E-R per gestire un archivio con informazioni su Musei, Opere e Artisti.
7 Disegna il diagramma E-R per gestire un archivio con informazioni su Aule e Lezioni.
8 Disegna il diagramma E-R per gestire un archivio con informazioni su Opere, Cantanti e Spettacoli.
9 Disegna il diagramma E-R per gestire un archivio con informazioni su Comuni, Regioni e Province.
10 Disegna il diagramma E-R per gestire un archivio con informazioni su Calciatori, Squadre e Partite.

ESERCIZI DI PROGETTAZIONE CONCETTUALE

Produci per le seguenti situazioni uno schema Entità-Relazione, e successivamente uno schema relazionale, indicando le eventuali ipotesi aggiuntive.

1 Gestione di dati su incidenti d'auto

Si vogliono gestire i dati di interesse di una compagnia di assicurazione ramo RCA creando una base di dati su clienti, auto e incidenti:

- per ogni cliente interessano codice fiscale (che lo identifica), nome e indirizzo mentre per le auto solamente la targa e modello;
- per gli incidenti è necessario memorizzare le auto assicurate coinvolte, l'ammontare del danno e la percentuale di colpa.

Si ipotizza che un'automobile abbia un solo proprietario.

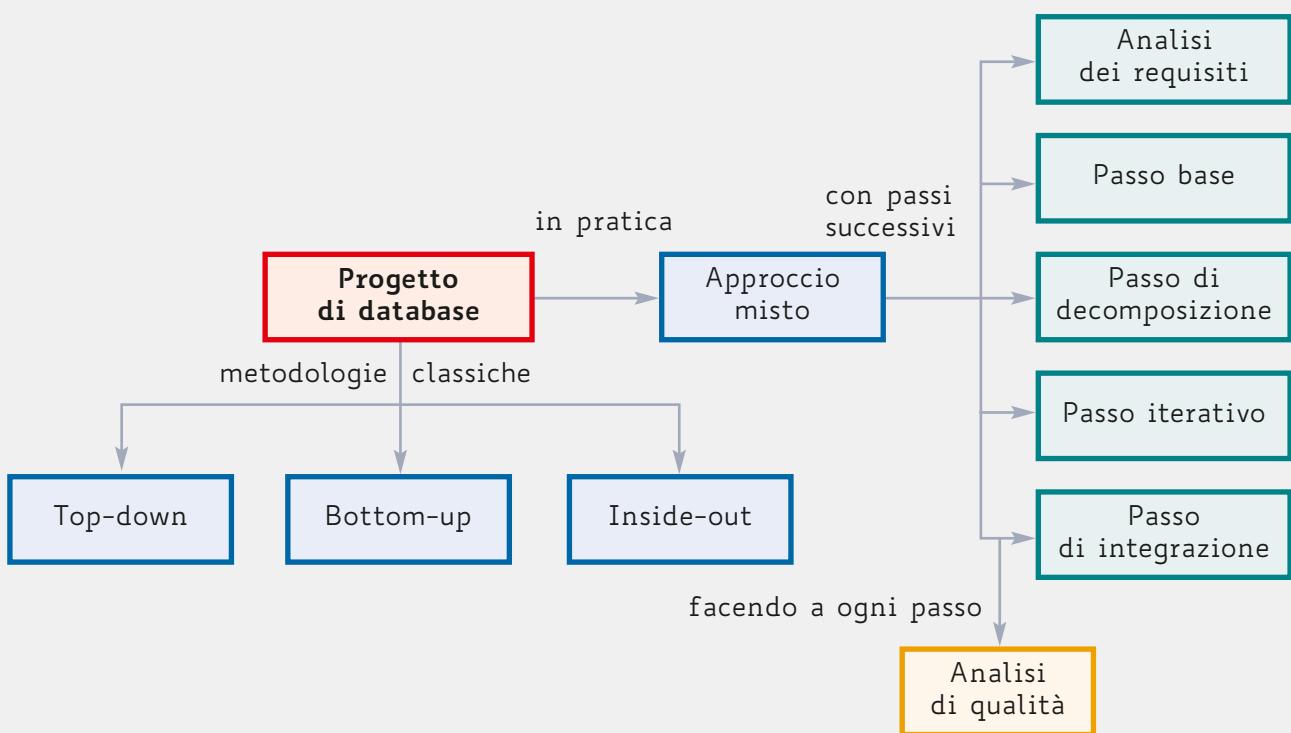
2 Azienda sanitaria locale

Si vuole costruire una base di dati per un'azienda sanitaria locale, tenendo traccia delle seguenti informazioni:

- gli ospedali sono caratterizzati da un codice identificativo, dal nome e dall'indirizzo;
- i dipendenti di ciascun ospedale sono caratterizzati dalla matricola (univoca all'interno dell'ospedale), dal cognome, dal nome, dalla data di nascita, dall'indirizzo e dal numero di figli a carico.

I dipendenti sono suddivisi in medici (di cui si conosce l'elenco delle specialità conseguite), impiegati amministrativi (di cui si conosce la mansione) e infermieri.
Un ospedale è suddiviso in reparti, caratterizzati da un codice, un nome, il numero di posti letto disponibili.

3 Gestione di dati sulla stagione operistica

Si vuole costruire una base di dati per la gestione della stagione concertistica di Como, che prevede una serie di concerti ognuno dei quali ha un codice, un titolo e una descrizione, ed è composto da un insieme di pezzi musicali, dove:

- ogni pezzo ha un codice, un titolo e uno o più autori (ciascuno con codice e nome) e uno stesso pezzo può essere rappresentato in diversi concerti;
- ogni concerto è eseguito da un'orchestra, che ha un nome, un direttore e un insieme di orchestrali;
- ogni orchestrale ha una matricola, nome e cognome, suona uno o più strumenti e può partecipare a più orchestre;
- ogni concerto è tenuto in una certa data, in una sala che ha un codice, un nome e una capienza.

Tecniche di progettazione dei diagrammi E-R

IN QUESTA LEZIONE IMPAREREMO...

- a padroneggiare le tecniche di progettazione dei database
- a utilizzare uno schema operativo per realizzare i diagrammi E-R

MAPPA CONCETTUALE

didattica inclusiva

Strategia di progettazione

Come per lo sviluppo degli algoritmi, anche per affrontare la progettazione di database complessi è opportuno adottare una strategia di progettazione che definisca le modalità con le quali procedere nello sviluppo.

Le metodologie principali sono tre:
- strategia top-down: si parte da uno schema iniziale molto astratto ma completo, che viene successivamente raffinato fino ad arrivare allo schema finale;
- strategia bottom-up: si suddividono le specifiche in modo da sviluppare semplici schemi parziali ma dettagliati, che poi vengono integrati tra loro;

– strategia inside-out: lo schema si sviluppa "a macchia d'olio", partendo dai concetti più importanti, aggiungendo quelli a essi correlati, e così via.

Ciascuna strategia ha pregi e difetti e si presta per specifiche situazioni di modellazione:
– nel metodo top-down non è inizialmente necessario specificare i dettagli, ma occorre avere sin dall'inizio una visione globale del problema, e ciò non sempre è possibile in casi complessi;
– nel bottom-up si procede con una ripartizione delle attività, che vengono inizialmente analizzate e sviluppate singolarmente, e sono poi seguite da una fase di integrazione;
– il metodo inside-out è simile alla "metodologia a spirale" (già utilizzata nella progettazione dei sistemi informatici) e non necessita di integrazione dato che consiste in una iterazione che richiede a ogni passo di esaminare tutte le specifiche per trovare i concetti non ancora rappresentati.

Nella pratica, nessuna di queste strategie viene utilizzata singolarmente e si privilegia un "approccio misto", nel quale si individuano i concetti principali, si realizza uno schema scheletro che contiene solamente i concetti più importanti e, sulla base di questo, si può decomporre il progetto in sottoparti che vengono singolarmente raffinate per poi essere integrate nello schema finale.

Schema operativo per i diagrammi E-R

Riportiamo in modo schematico i principali passi da seguire nella metodologia della "strategia mista".
1 Analisi dei requisiti: in questa fase si analizzano i requisiti per eliminare le ambiguità e per costruire un glossario dei termini.
2 Passo base: viene definito lo schema scheletro con l'indicazione dei concetti più rilevanti.
3 Passo di decomposizione: quando è necessario si procede con la decomposizione dei requisiti prendendo come riferimento i concetti sintetizzati nello schema scheletro.
4 Passo iterativo: viene ripetuto finché non si ottiene uno schema dettagliato; consiste nel raffinamento dei concetti presenti sulla base delle loro specifiche aggiungendo le specifiche non ancora descritte.
5 Passo di integrazione: nei casi complessi il passo finale è quello che, prendendo come riferimento lo schema scheletro, permette di integrare i vari sottoschemi in uno schema complessivo.
6 Analisi di qualità: è una operazione che deve essere presente in tutte le fasi per verificare la qualità dello schema e modificarlo in itinere per evitare di individuare gli errori solo alla fine del processo di modellizzazione e dover ripetere dall'inizio il progetto.

Un esempio completo: corsi estivi di recupero

SITUAZIONE

Si vuole realizzare una base di dati per i corsi di recupero estivi, in cui verranno rappresentati i dati dei partecipanti ai corsi e dei docenti.
I partecipanti sono alunni che hanno avuto il debito in quella materia, oppure studenti esterni che hanno richiesto il passaggio di istituto: vengono identificati da una matricola e si vuole memorizzare il codice fiscale, il cognome, l'età, il sesso, la classe e la sezione di appartenenza, il numero di telefono, i corsi che stanno frequentando e/o hanno frequentato, il giudizio finale e l'eventuale scuola di provenienza.
I corsi hanno un codice, un titolo, una classe alla quale sono rivolti, un luogo dove sono tenuti e possono avere varie edizioni con date di inizio e fine e numero di partecipanti diversi.
Per gli insegnanti verrà indicato il cognome, l'età e il luogo dove sono nati, il nome del corso che tengono e di quelli che hanno tenuto nel passato e tutti i loro recapiti telefonici.
I docenti possono essere dipendenti interni della scuola oppure collaboratori esterni.

SOLUZIONE

Per risolvere questa situazione applichiamo lo schema operativo sopra proposto tenendo presente le diverse fasi di ogni passo descritte nella lezione precedente: dato che il problema non è molto complesso non è necessario ripetere iterativamente le operazioni, ma con un unico affinamento arriviamo alla soluzione desiderata.

1. Per prima cosa analizziamo il testo per ristrutturare le frasi eliminando le ambiguità:

Sinonimi
– insegnanti, docenti
– alunni, partecipanti ai corsi, studenti

Omonimi
– luogo
– dove si tengono i corsi
– luogo di nascita

2. Costruiamo ora il glossario dei termini, fornendo per ogni termine una breve descrizione, i sinonimi e gli altri termini con cui esiste un legame logico: è molto utile per facilitare la comprensione e la precisazione dei termini.

TERMINE	DESCRIZIONE	SINONIMI	LEGAME
Partecipante	Partecipante ai corsi Interno oppure esterno alla scuola	Studente Alunno	Corso
Docente	Docente dei corsi Possono essere collaboratori esterni	Insegnante	Corso, Materia
Corso	Corsi offerti di una materia Possono avere varie edizioni Sono indirizzati ad alunni specifici		Docente, Partecipante
Materia	Oggetto del corso		Corso, Docente

3. Procediamo con la decomposizione del testo in gruppi di frasi omogenee.

Frasi di carattere generale

Si vuole realizzare una base di dati per i corsi di recupero estivi, in cui verranno rappresentati i dati dei partecipanti ai corsi e dei docenti.

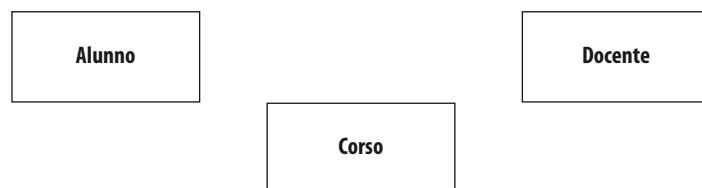
Frasi relative ai partecipanti

I partecipanti sono:
– alunni che hanno avuto il debito in quella materia (giudizio sospeso);
– studenti esterni che hanno richiesto il passaggio di istituto.

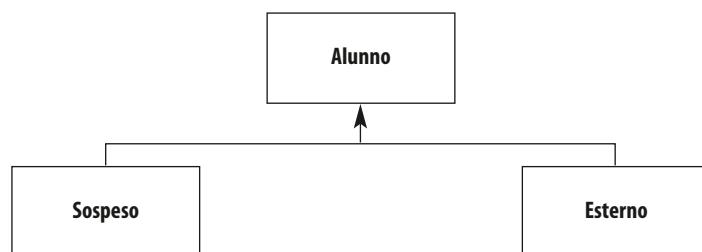
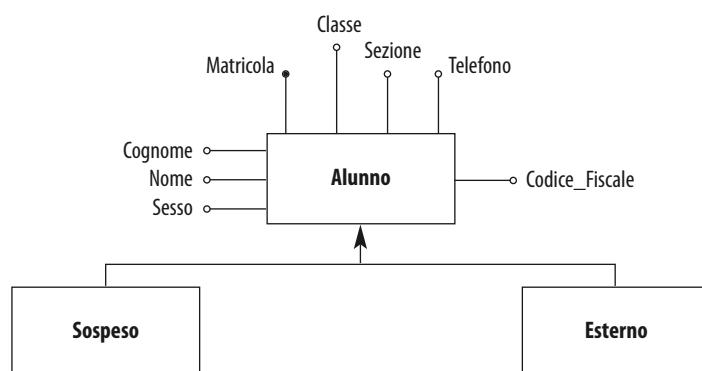
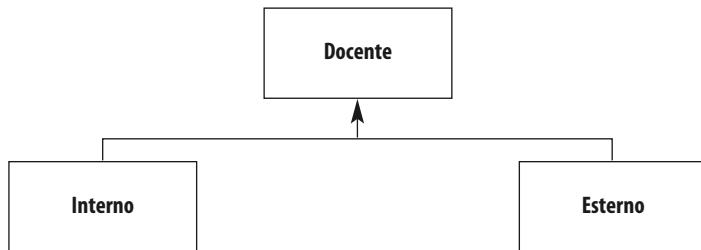
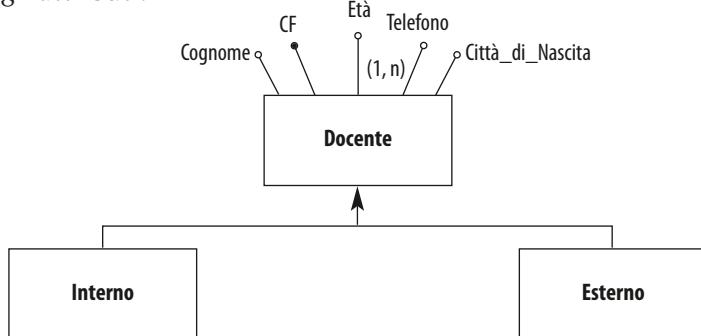
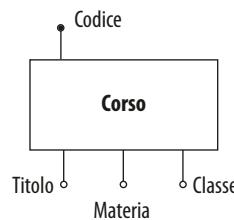
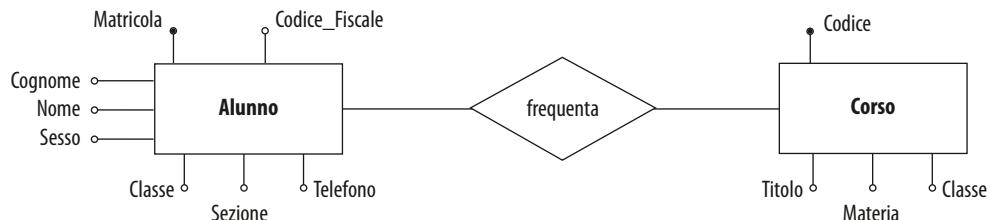
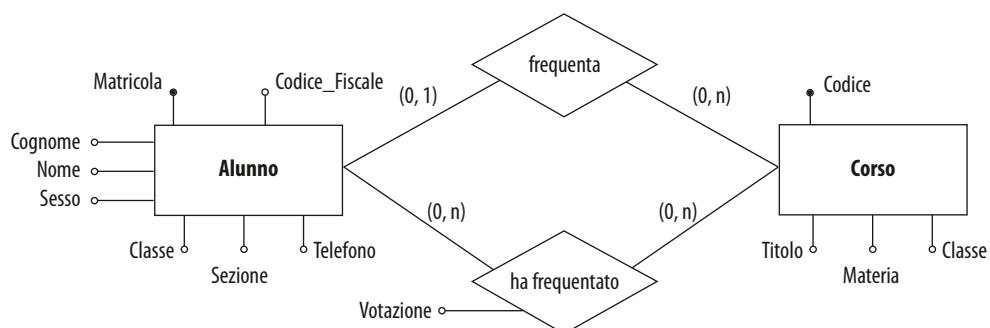
I partecipanti vengono identificati da una matricola e per essi si vuole memorizzare:
– il codice fiscale, il cognome, l'età, il sesso, la classe e la sezione di appartenenza, il numero di telefono, i corsi che stanno frequentando e/o hanno frequentato e il giudizio finale e l'eventuale scuola di provenienza.

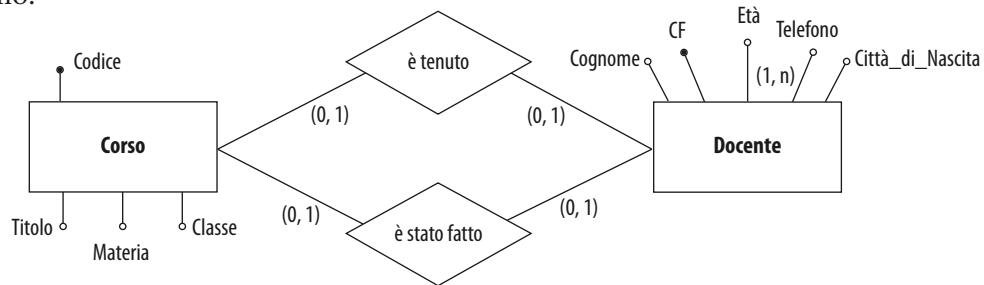
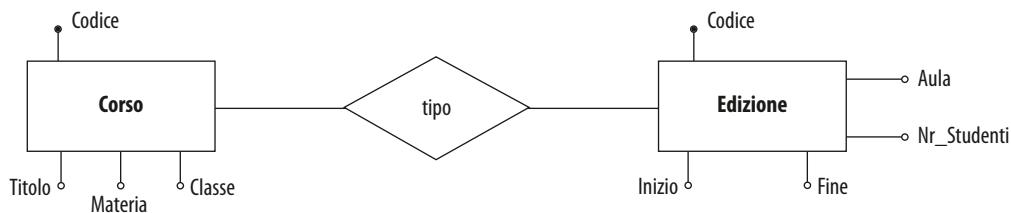
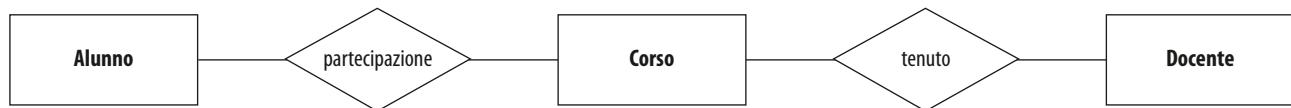
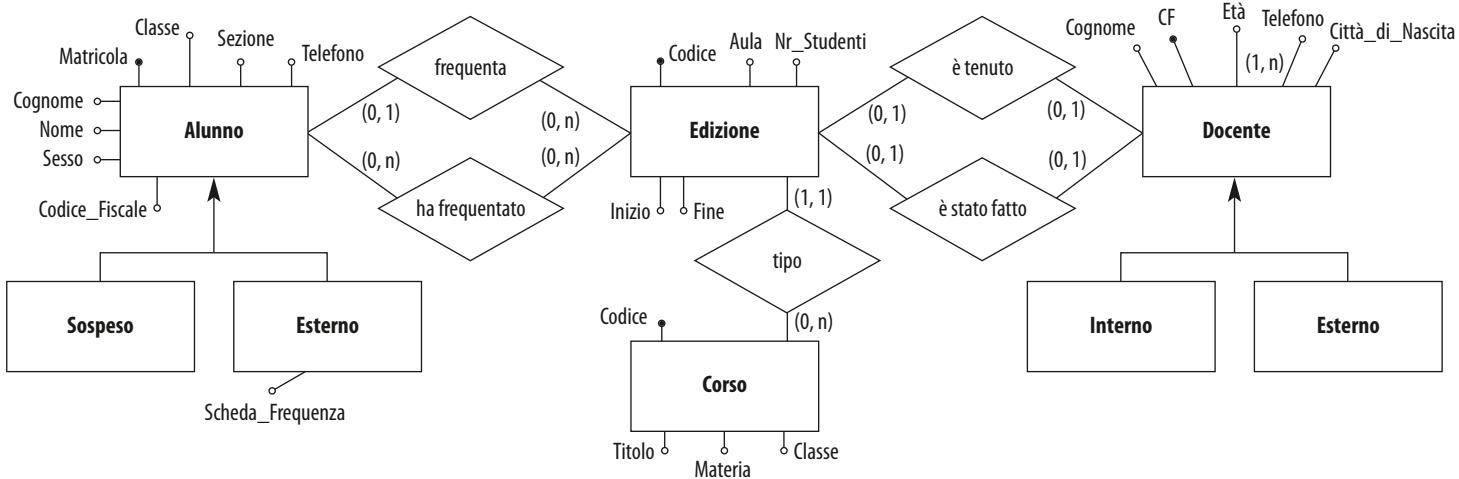
Frasi relative ai docenti

I docenti possono essere:
– dipendenti interni della scuola;
– collaboratori esterni.

Per i docenti indichiamo:
– il cognome, l'età e il luogo dove sono nati, il nome del corso che seguono, quelli che hanno tenuto nel passato e tutti i loro recapiti telefonici.

Frasi relative ai corsi

I corsi hanno un codice, un titolo, una classe alla quale sono rivolti, un luogo dove sono tenuti e possono avere varie edizioni con date di inizio e fine e numero di partecipanti diversi.
I partecipanti ai corsi hanno avuto il debito in quella materia.

4. A questo punto, possiamo definire le entità principali, che sono:
– Alunno;
– Docente;
– Corso.

La prima bozza dello schema scheletrico è la seguente:

Il processo di decomposizione consiste nel prendere in considerazione una alla volta ciascuna entità indicando le gerarchie e gli attributi che sono emersi dell'analisi del testo.

Alunno: è composto da due sotto-entità distinte con una gerarchia totale ed esclusiva, dato che può essere Sospeso o Esterno.

Aggiungiamo gli attributi.

Docente: analogamente all'alunno, anche per il docente abbiamo una gerarchia totale ed esclusiva con due sotto-entità, dato che può essere Interno o Esterno.

Aggiungiamo gli attributi.

Corso: non ci sono gerarchie per l'entità Corso e la riportiamo già con i suoi attributi.

5. Il passo successivo consiste nell'aggiungere le relazioni tra le entità, a partire da Alunno e Corso.

Dato che un Alunno può aver frequentato in passato altri Corsi è necessario indicare anche questo tipo di relazione.

Aggiungiamo ora le relazioni tra le entità Corso e Docente.

Analogamente al caso precedente, il Docente tiene un Corso oppure può averlo tenuto nel passato, quindi otteniamo.

Un Corso può poi avere diverse edizioni, quindi introduciamo una relazione che ci permette di esprimere questa situazione.

Completiamo ora lo schema scheletrico è aggiungendo le relazioni tra le entità.

6. Il passo di integrazione consiste nel "mettere assieme" i diagrammi parziali ottenendo lo schema finale.

Possiamo osservare come lo schema E-R finale, "apparentemente complesso", sia stato di fatto ottenuto mediante un procedimento top-down con aggregazione finale delle sotto-parti.

METTITI ALLA PROVA

Progetto schema E-R • Modifica di specifiche

Modifica opportunamente lo schema in base al cambiamento delle seguenti specifiche:

1. Il programma del corso è composto da un insieme di lezioni che possono essere tenute in aula oppure in laboratorio;
2. I docenti devono avere l'abilitazione specifica per tenere un certo corso.

AREA DIGITALE

UML vs grafica a rombi

VERIFICA... le competenze

AREA DIGITALE

Esercizi per il recupero e il rinforzo

ESERCIZI DI PROGETTAZIONE CONCETTUALE

Delle seguenti situazioni produci uno schema Entità-Relazione, e successivamente uno schema relazionale, indicando le eventuali ipotesi aggiuntive.

1 Agenzia automobilistica

Si progetti lo schema E-R per descrivere una situazione tra automobili e rispettivi proprietari.
Una automobile è descritta da una targa, un modello, una marca, e un colore e ogni auto ha un unico proprietario attuale, descritto da codice fiscale, nome e cognome.
Un proprietario può possedere più automobili e, nel tempo, può cambiare automobili: si vuole memorizzare anche la durata del periodo di possesso di ciascun autoveicolo per ciascun proprietario.

2 Pubblicazione

Si progetti lo schema E-R per descrivere una situazione tra autori e pubblicazioni.
Una pubblicazione ha un titolo, una lista di autori, di cui si conoscono solo il nome e il cognome, e una data di pubblicazione, appartiene a una singola categoria e può contenere dei riferimenti bibliografici ad altre pubblicazioni.
Le categorie possono essere libro, articolo su rivista, tesi di laurea su un argomento specifico e discussa in una singola università.

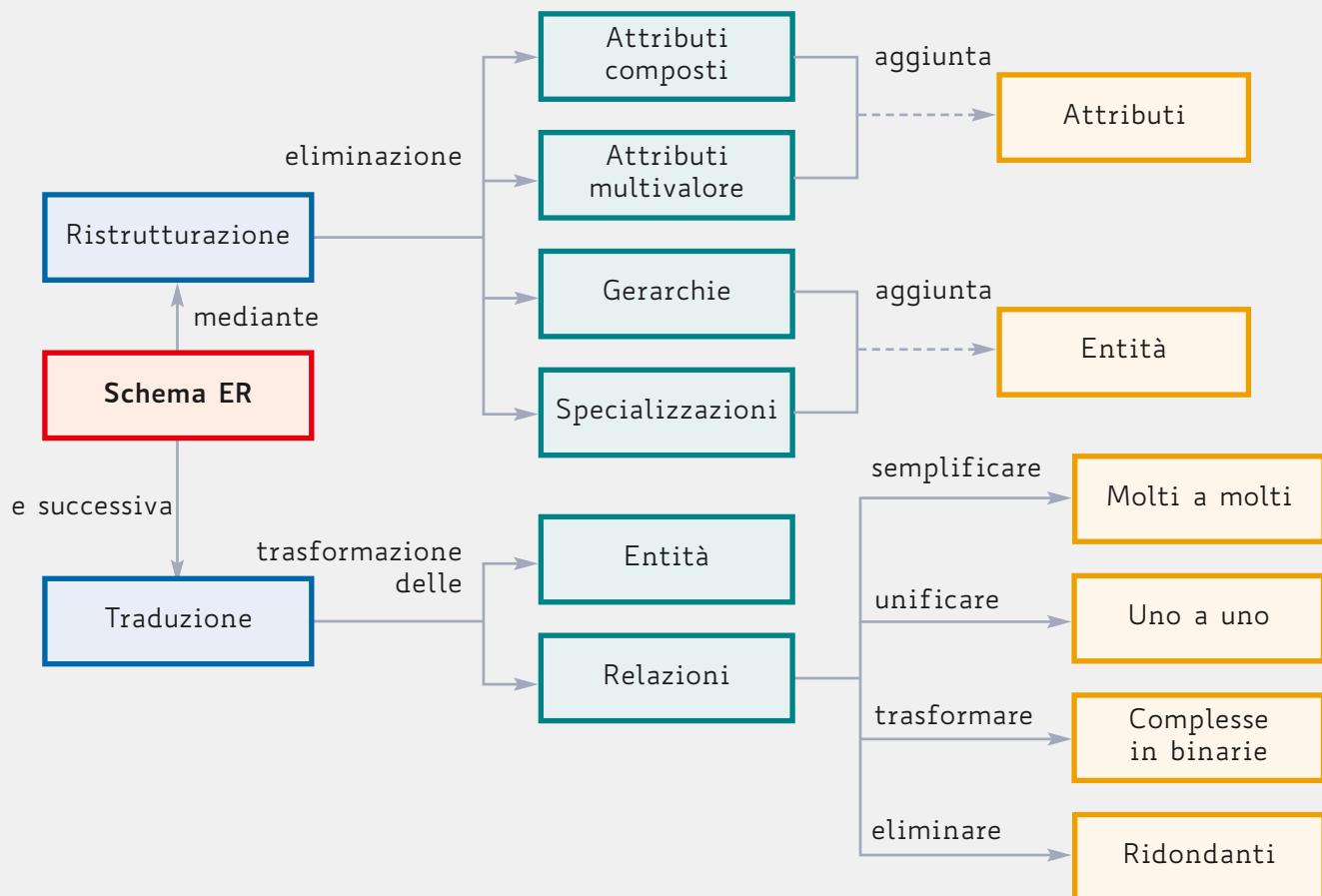
3 Agenzia immobiliare

Si progetti lo schema E-R per descrivere una agenzia immobiliare.
In particolare, si considerino solo gli immobili in vendita, identificati da un codice, classificati per tipo (appartamento, villa, ...) con le eventuali pertinenze (garage, cantina, giardino ecc.).
Il/i proprietario/i stabilisce un prezzo di vendita comunicando la scheda catastale dalla quale è direttamente deducibile la superficie, il numero di vani e la loro tipologia (soggiorno, studio, camera, corridoio ecc.).
Il/I proprietario/i è identificato dal codice fiscale e sono noti i suoi dati anagrafici.

4 Incidenti automobilistici

Si progetti lo schema E-R per memorizzare informazioni su incidenti automobilistici.
Le automobili coinvolte (una o più per sinistro) sono identificate dalla targa e per ciascuna di esse sono noti la marca, il modello, la cilindrata, la potenza, il/i proprietario/ e la compagnia presso cui è assicurata.
I dati riguardanti il proprietario, identificato tramite il suo codice fiscale, sono il cognome, nome, la residenza e lo stato civile.
Per ogni sinistro è necessario memorizzare la data e la località in cui è avvenuto, il nome della strada, le automobili coinvolte, una descrizione del danno riportato da ciascuna auto e la presenza o meno di feriti.

5 Foto digitali

Si progetti lo schema E-R per memorizzare una raccolta di fotografie presenti in un computer.
A ogni foto è dato un nome ed è archiviata in una specifica cartella: i dati da memorizzare sono la data in cui è stata scattata, il luogo in cui è stata scattata e un codice che la identifica.
Se in una foto sono presenti persone queste devono essere identificate da un codice e si deve conoscere il nome e il cognome e, se è disponibile, la liberatoria per l'utilizzo della fotografia.
Se una foto è collegata a un evento, come ad esempio un matrimonio, una gita, una festa ecc., tale evento deve essere classificato assegnandogli un nome per la sua completa identificazione e, inoltre, di tale evento si deve memorizzare la data di inizio e la data di fine oltre che alle persone che sono presenti in tutte le fotografie a esso relative.

Dal modello E-R allo schema logico

IN QUESTA LEZIONE IMPAREREMO...

- a raffinare un modello E-R
- a descrivere il modello logico con lo schema relazionale

MAPPA CONCETTUALE

didattica inclusiva

Schema ER
mediante
Ristrutturazione
eliminazione
Attributi composti
Attributi multivalore
aggiunta
Attributi
Gerarchie
Specializzazioni
aggiunta
Entità
e successiva
Traduzione
trasformazione delle
Entità
Relazioni
semplificare
Molti a molti
unificare
Uno a uno
trasformare
Complesse in binarie
eliminare
Ridondanti

Il modello logico

Il modello logico del database (o schema logico) è lo strumento che viene utilizzato come input per la progettazione fisica dei database: deve quindi avere il massimo livello di dettaglio e precisione possibile.

Il modello logico dota i dati di una struttura utile per realizzare, semplificare e ottimizzare le operazioni di archiviazione, interrogazione e manipolazione dei dati.

Deve quindi contenere tutte le informazioni necessarie per definire fisicamente le tabelle, riportando la descrizione puntuale e completa del significato di ogni dato che viene memorizzato in esse.

Tale livello di precisione viene raggiunto progressivamente durante le varie fasi di progettazione acquisendo la conoscenza approfondita della applicazione che si sta realizzando.

Il modello logico, pertanto, non aggiunge nuove informazioni allo schema concettuale ma le completa e le trasforma in un formato "efficiente": i due schemi sono quindi equivalenti dal punto di vista della loro capacità informativa.

Dato che deve essere la base per la progettazione fisica del database, esiste un modello logico per ogni specifico tipo di DBMS: noi utilizzeremo il modello relazionale e il nostro modello logico consisterà nella definizione delle tabelle relazionali (o schema relazionale).

Operativamente, le tabelle relazionali si ottengono dallo schema concettuale mediante una operazione che prende il nome di mappatura (mapping): questa fase della progettazione consiste nell'applicazione di semplici regole di derivazione che trasformeranno ogni elemento dello schema concettuale nello schema logico, definendo il singolo schema relazionale, tabella per tabella.

Un semplice schema relazionale come il seguente:

Alunni (matricola(pk), cognome , nome, …, scuola(fk))

può essere considerato come schema logico: è tuttavia sempre preferibile realizzare un modello logico preciso (o completo) per rendere più semplice la progettazione fisica – in tutte le possibili alternative – di DBMS relazionali che saranno utilizzati nella pratica, aggiungendo cioè quelle informazioni grazie alle quali successivamente lo si possa utilizzare indifferentemente per MySql oppure Oracle, per Access oppure Sybase, ecc).

Un modello logico preciso comprende:
- per ogni entità, l'elenco completo degli attributi;
- per ogni entità, l'indicazione esplicita della chiave primaria e di eventuali chiavi alternative;
- per ogni entità, l'indicazione esplicita di eventuali chiavi esterne;
- per ogni attributo, l'indicazione esplicita di opzionalità o obbligatorietà;
- per ogni attributo, l'indicazione esplicita del tipo di dati (data *type*), che ne specifichi il formato e la lunghezza, con la segnalazione dei valori ammessi;
- per ogni relazione, l'indicazione della molteplicità minima e massima in entrambe le direzioni;
- per ogni relazione, l'indicazione delle regole di integrità referenziale applicabili (che verranno descritte nella lezione 12).

ESEMPIO

Un esempio di modello logico preciso per l'esempio precedente è il seguente:

TABELLA ALUNNI				
Attributo/campo	**Tipo**	**Dimensione**	**Valori**	**Note**
Matricola	Numerico	8	Autoincrement	pk
Cognome	Stringa	40	Not null	
Nome	Stringa	30	Not null	
...				
Scuola	Numerico	12	Codifica del MIUR	fk

Durante la progettazione logica si definiscono inoltre gli schemi esterni (viste) per le specifiche applicazioni.

Dallo schema E-R allo schema logico

Il passaggio dallo schema E-R allo schema logico viene effettuato mediante due fasi:
– la fase di ristrutturazione del diagramma E-R;
– la fase di traduzione nello schema logico e nelle tabelle relazionali.

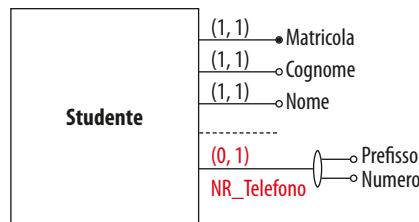
Ogni fase prevede particolari attività da svolgere in sequenza, come una check list di controlli, che di fatto non sono immuni da insidie e possono produrre errori: ricordiamo che questa fase porta da uno schema concettuale a uno schema logico e, per tutta la durata della trasformazione, è necessario sempre tenere presente il problema da risolvere nella sua totalità, per non perdere, "strada facendo", la visione completa del progetto.

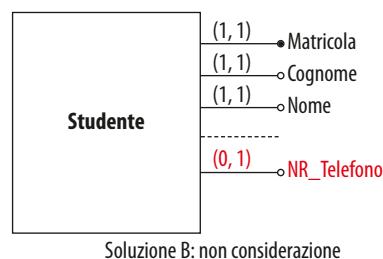
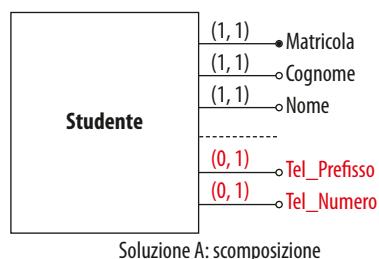
Ristrutturazione del diagramma E-R

Una volta che lo schema E-R è stato completamente definito, si procede al suo affinamento (ristrutturazione) applicando le regole base di modellizzazione che verranno descritte di seguito e che ci permetteranno di eliminare tutti i costrutti che non possono essere direttamente rappresentati nel modello relazionale.

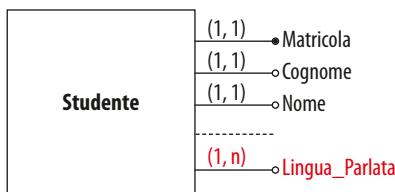
1. PARTECIPAZIONE DELLE ENTITÀ ALLE RELAZIONI

Le entità non possono essere modellate in modo da essere scollegate da altre entità, altrimenti, quando il modello viene trasformato in un modello relazionale, non ci sarà alcun modo per collegare quella tabella con le altre.

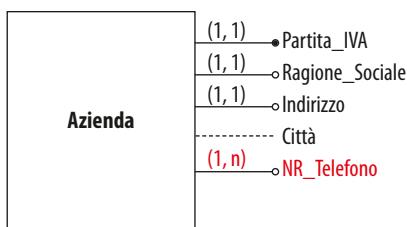
> L'eccezione a questa regola consiste in un database formato da una singola tabella, come per esempio il carrello elettronico utilizzato nei siti di commercio elettronico.

2. ELIMINAZIONE DEGLI ATTRIBUTI COMPOSTI: AGGIUNTA DI ATTRIBUTI

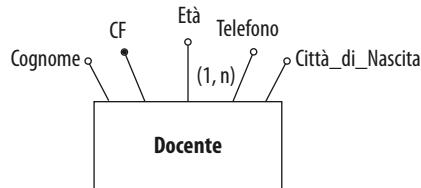
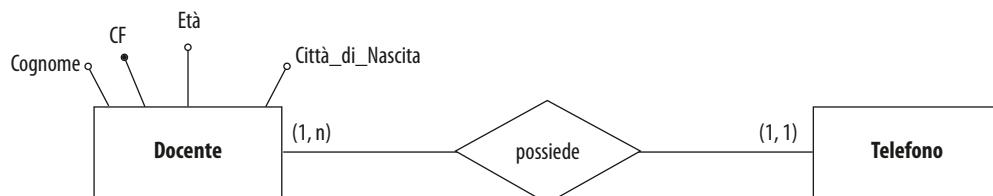
Nel caso fosse presente un attributo composto si può procedere in due modi alternativi:
– considerare tutti i sotto-attributi come attributi;
– eliminare i sotto-attributi e considerare l'attributo composto come un attributo semplice.

ESEMPIO

Nell'entità Studente è presente come attributo composto il NR_Telefono.

Le due soluzioni alternative per rappresentarlo sono le seguenti.

3. ELIMINAZIONE DEGLI ATTRIBUTI MULTIVALORE: AGGIUNTA DI ENTITÀ

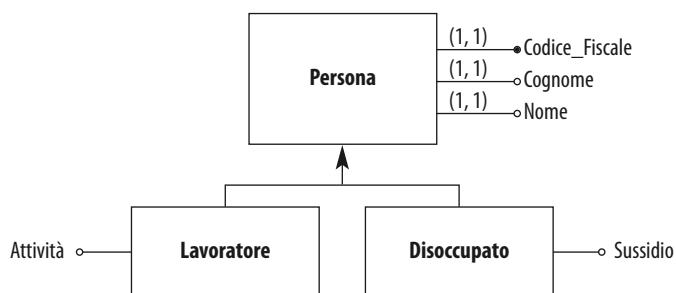
Gli eventuali attributi multivalore presenti devono essere "promossi" a entità: si crea una nuova entità che contiene i valori dell'attributo e la si collega all'entità che possedeva l'attributo mediante una nuova relazione uno-a-molti o molti-a-molti, a seconda dei casi.

ESEMPIO

a) Uno Studente può parlare più di una lingua.

Il diagramma ristrutturato, dove viene aggiunta l'entità Lingua, diviene il seguente:

b) Una Azienda può possedere più numeri di telefono.

Il diagramma ristrutturato, dove viene aggiunta l'entità Telefono, diviene il seguente:

c) Anche nell'ultimo esempio della lezione 7 c'era una situazione di questo tipo.

Dato che Telefono è un attributo multivalore, è necessario introdurre una nuova entità nella fase di ristrutturazione.

4. ELIMINAZIONE DELLE GERARCHIE E SPECIALIZZAZIONI

La gerarchia di generalizzazione deve essere trasformata in una unica entità in quanto non è supportata dal modello relazionale.

È possibile ristrutturare il diagramma in due modi differenti:

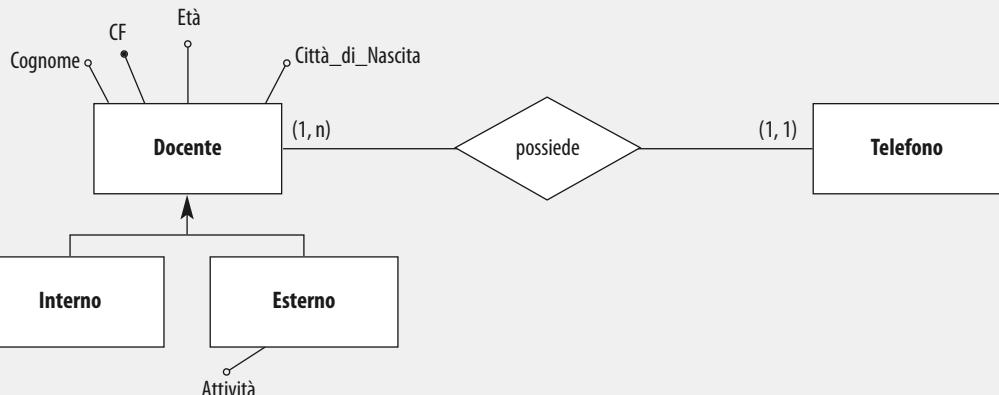
a) eliminazione dei figli (collasso dei figli), aggiungendo uno o più attributi nella entità padre;

b) eliminazione del padre (esplosione dei figli): completando ciascun figlio con gli attributi del padre.

ESEMPIO

La seguente situazione che mostra due entità figlie per l'entità Persona

può essere trasformata nelle due seguenti alternative:

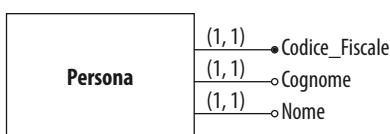
Nel caso **A** abbiamo una sola entità nella quale sono stati aggiunti gli attributi che rimarranno vuoti a seconda del caso della singola istanza; nel caso **B** abbiamo invece due entità distinte.

Generalmente viene scelta la prima soluzione in quanto, riducendo il numero delle entità, si ridurranno anche il numero delle tabelle nel database.

METTITI ALLA PROVA

➜ Eliminazione delle gerarchie • Eliminazione e specializzazioni

In riferimento all'esempio precedente, relativo alla entità Docente, elimina le gerarchie proponendo entrambe le soluzioni, cioè "collassando" verso il padre o "esplodendo" verso i figli.

Traduzione del modello E-R nel modello relazionale

Dopo aver applicato al modello E-R le regole di ristrutturazione appena descritte si ottiene quello che viene anche chiamato schema E-R ristrutturato, pronto per essere trasformato nello schema relazionale.
In questa fase, che prende il nome di fase di traduzione, vengono applicate le regole di trasformazione di entità, attributi e associazioni dello schema E-R in relazioni del modello relazionale. Esaminiamo queste regole una a una.

1. TRASFORMAZIONE DELLE ENTITÀ

Per ogni entità viene generata una tabella indicando il suo schema relazionale, dove viene riportato un campo per ogni attributo dell'entità.

ESEMPIO

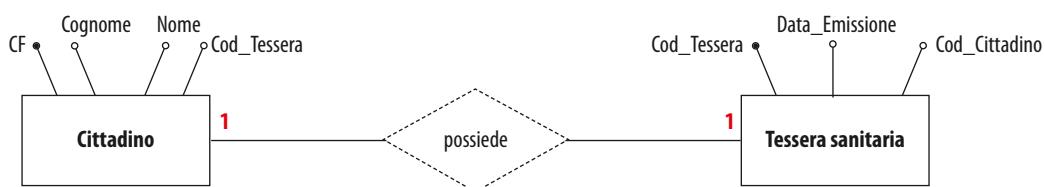
Il seguente diagramma E-R, riferito alla entità Persona

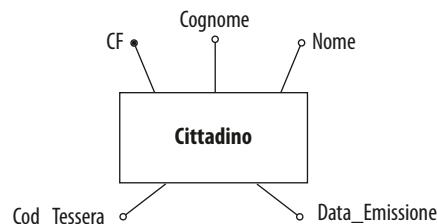
viene tradotto nel seguente schema relazionale:

Persona (Codice_Fiscale(pk), Cognome, Nome)

In caso di presenza di attributi calcolati va aggiunto un vincolo di integrità che specifica il metodo di calcolo. Un caso abbastanza frequente di attributo calcolato è l'attributo età: questo viene calcolato come differenza di due date, la data odierna e la data di nascita di una persona, e se manca il valore di una di esse il suo valore deve essere null.

2. TRASFORMAZIONE DELLE RELAZIONI

Si effettua la traduzione delle associazioni, in base al numero di entità partecipanti e alla loro cardinalità.

a) Unificare le relazioni uno-a-uno

Due entità legate da una relazione 1, 1 possono essere ridotte a un'unica entità che contiene gli attributi sia della prima sia della seconda entità.

ESEMPIO

Consideriamo la seguente frase: "Un cittadino possiede una tessera sanitaria".
Se descriviamo questa situazione con uno schema E-R possiamo ottenere:

La relazione è del tipo 1, 1, in quanto un cittadino può possedere una sola tessera sanitaria e ogni tessera sanitaria è unica per ogni cittadino: quindi la relazione è eliminabile e otteniamo un'unica entità con tutti gli attributi.

Esistono situazioni dove unificare le relazioni 1, 1 potrebbe portare alla perdita di significatività di una entità in gioco: in questo caso si introduce una migrazione della chiave di una entità come chiave esterna alla seconda entità.

ESEMPIO

Nella seguente situazione la relazione tra le due entità è 1, 1 ma entrambe "meritano" di esistere, cioè è difficile decidere quale deve essere eliminata e inglobata nell'altra.

Abbiamo due possibilità:

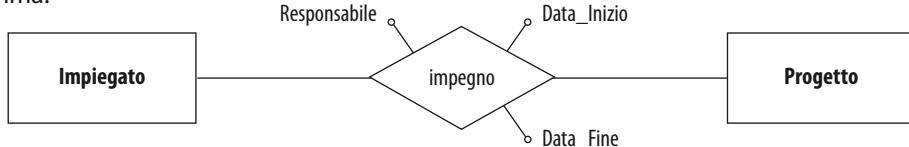
Soluzione a)	**Soluzione b)**
Stato (Nome_Stato(pk))	Presidente (Cognome(pk))
Presidente (Cognome (pk), Nome_Stato(fk))	Stato (Nome_Stato(pk), Cognome_Presidente(fk))

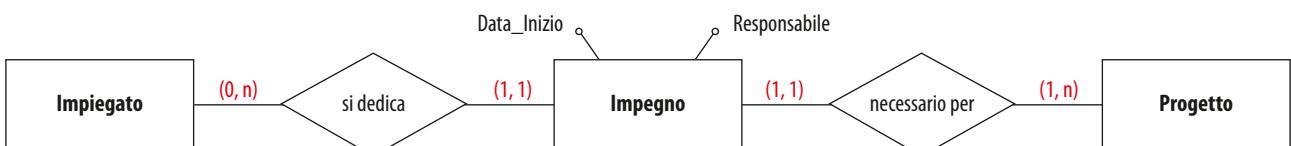
In questo esempio unificare l'entità tecnicamente "sarebbe una soluzione giusta", ma metodologicamente è sbagliata dato che, a livello concettuale, i concetti di Stato e Presidente sono stati separati e tale separazione deve persistere anche a livello logico.

b) Semplificare le relazioni molti-a-molti

Le relazioni n, n non possono essere usate nel modello dei dati perché non possono essere rappresentate nel modello relazionale: devono quindi essere risolte nelle fasi di ristrutturazione del modello E-R sostituendole con un'entità associativa (anche detta entità ponte) che va messa in relazione con le due entità originali.

ESEMPIO

Consideriamo la situazione seguente, in cui:

– gli impiegati possono essere impegnati in molti progetti;
– ogni progetto deve impegnare più di un impiegato.

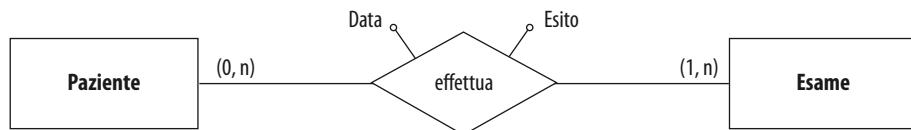
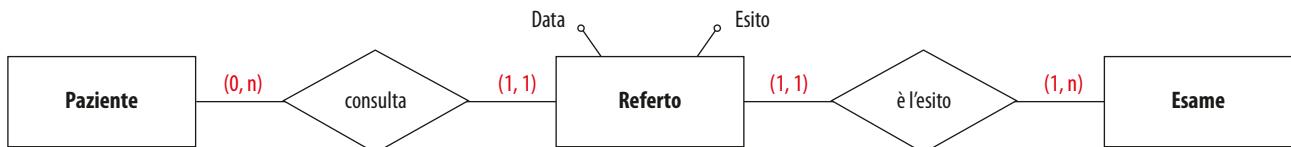
Siamo in presenza di una relazione n, n, descritta dal seguente diagramma E-R.

Oltre ai problemi implementativi, questa relazione presenta altre criticità: supponiamo di voler aggiungere un'informazione relativa agli impegni di un impiegato, per esempio il nome del responsabile che ha deciso gli abbinamenti tra gli impiegati e i progetti, la data di inizio dell'impegno e la data di fine.

Questi sono attributi della relazione e andrebbero quindi messi su di essa, come nel seguente diagramma:

Il primo passo per affinare il modello E-R consiste nel convertire la relazione Impegno in una nuova entità, che chiamiamo Impegno. Quindi, le entità originali Impiegato e Progetto vengono messe in relazione con questa nuova entità preservando la cardinalità e l'opzionalità delle relazioni originali.

La rappresentazione grafica di questo affinamento è illustrata nella figura seguente:

A questo punto, si aggiungono gli attributi e si definiscono le chiavi, in modo da ottenere uno schema relazionale come il seguente:

Impiegato (ID_Impiegato(pk), Cognome, Nome)
Progetto (ID_Progetto(pk), Descrizione)
Impegno (ID_Impiegato(pk), ID_Progetto(pk), Data_Inizio, Responsabile)

Vediamo un secondo esempio.

ESEMPIO

Consideriamo la situazione seguente, in cui:
- un paziente può effettuare molti esami;
- un esame è presente in più esami effettuati dal paziente.

Il diagramma E-R che segue rappresenta la situazione, che è analoga a quella dell'esempio precedente.

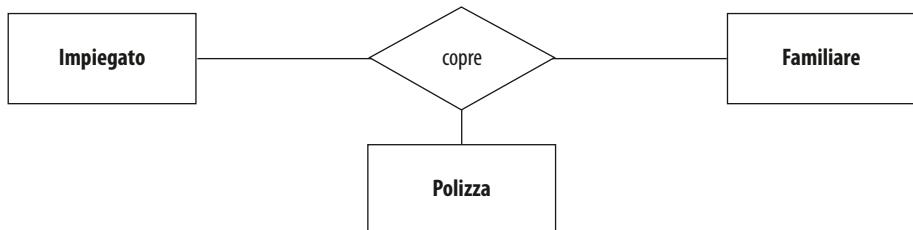
L'affinamento del diagramma è illustrato nella figura seguente, dove è stata introdotta l'entità Referto:

A questo punto, si aggiungono gli attributi e si definiscono le chiavi, in modo da ottenere uno schema relazionale come il seguente:

Paziente (ID_Paziente (pk), Cognome, Nome)
Esame (ID_Esame(pk), Descrizione)
Referto (ID_Paziente(pk), ID_esame (pk), Data, Esito))

METTITI ALLA PROVA

➜ Eliminazione relazioni 1, 1 • Semplificazione relazioni n, n

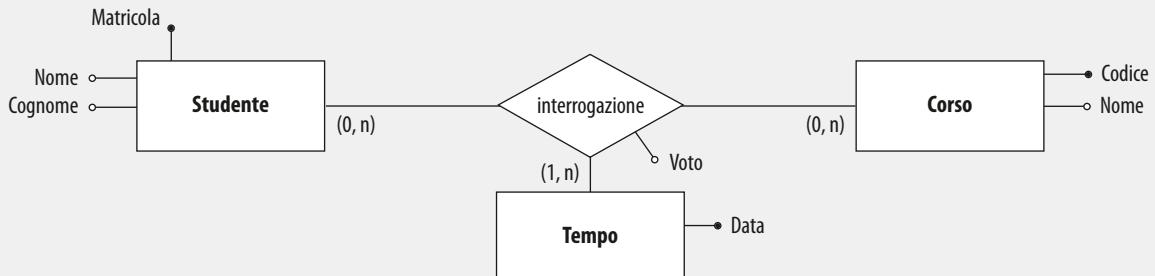
Realizza i diagrammi E-R delle seguenti situazioni e semplifica le eventuali relazioni n:n introducendo una entità associativa ed evidenziandone gli eventuali attributi.
- Autista-Autobus
- Autore-Libro
- Cantante-Canzone
- Docente-Materia
- Partita-Squadra

c) Eliminare le relazioni ternarie

Le relazioni complesse sono relazioni che coinvolgono più di due entità: non possono essere implementate direttamente nel modello logico e quindi devono essere risolte nella fase di ristrutturazione.

Per risolvere relazioni complesse si procede come l'introduzione di una nuova relazione, in modo da ottenere due relazioni binarie.

ESEMPIO

Ogni polizza è posseduta da un solo impiegato e copre i propri familiari.

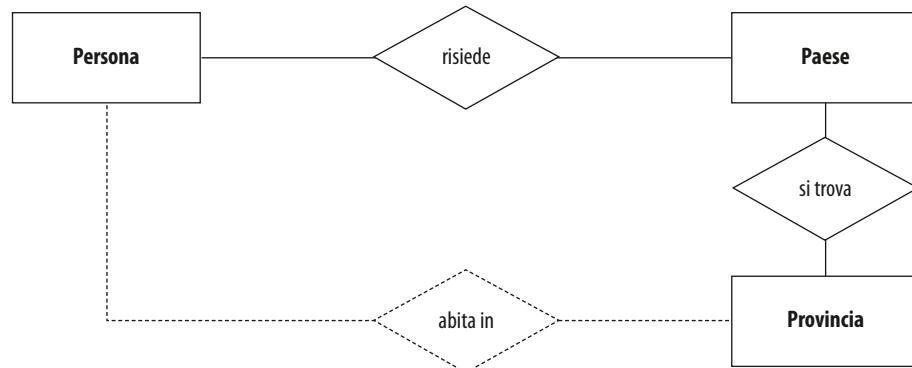
Introduciamo la relazione che lega la Polizza con l'Impiegato, cioè definisce l'Intestatario.

È evidente come due relazioni binarie siano migliori di una relazione ternaria.

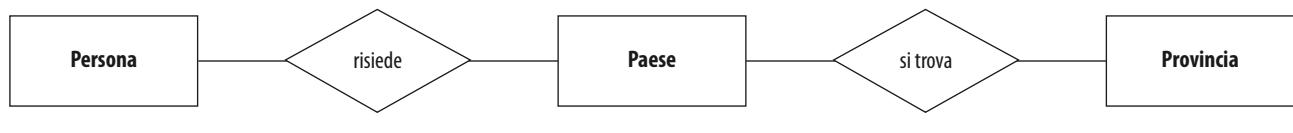
Data la seguente relazione ternaria:

trasforma lo schema E-R in modo da ottenere il seguente schema relazionale.

Studente (Matricola(pk), Nome, Cognome)
Corso (Codice(pk), Nome)
Tempo (Data(pk))
Interrogazione (Matricola(pk), Codice, Data, Voto)

d) Eliminare le relazioni ridondanti

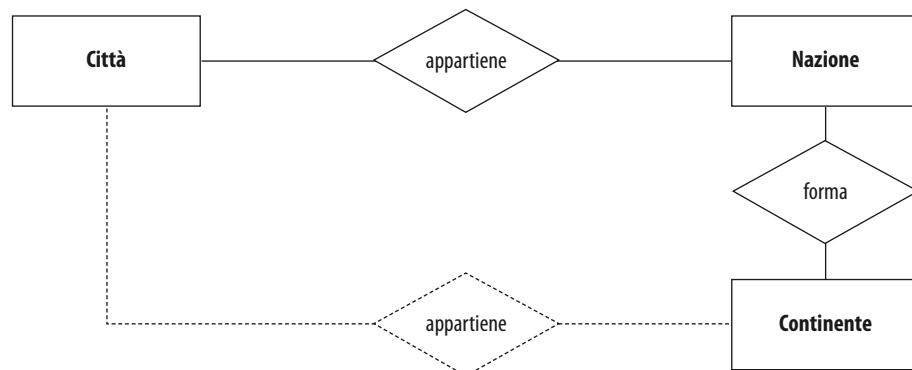
Una relazione ridondante è una relazione definita tra due entità e che è equivalente nel significato a un'altra relazione tre le stesse due entità che passa attraverso un'entità intermedia.

Vediamo un primo esempio.

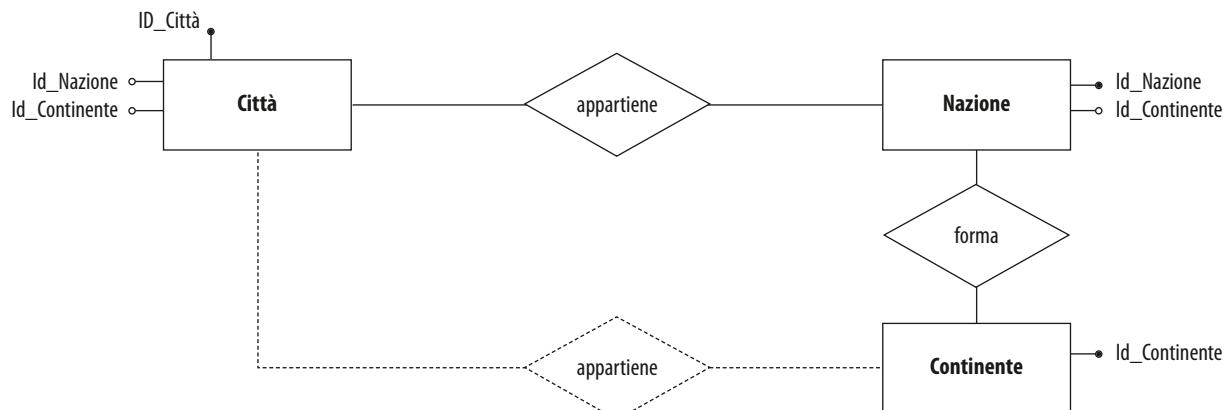
ESEMPIO

Consideriamo la situazione seguente:
- Una persona abita in una provincia.
- Una persona risiede in un paese.
- Il paese si trova in una provincia.

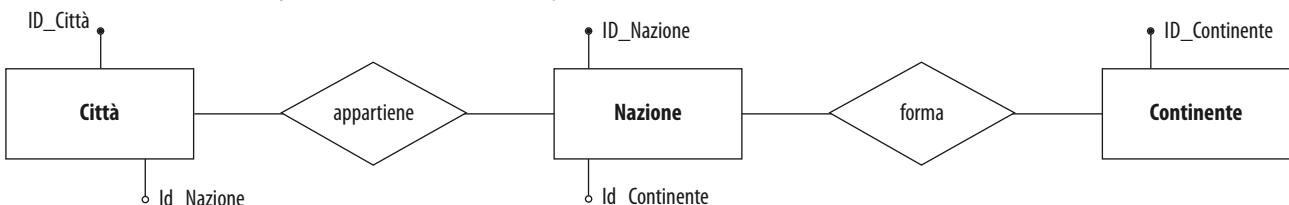
Partendo da queste frasi, ricaviamo il diagramma E-R.

Nella figura è presente una relazione ridondante tra Persona e Provincia, data dalla prima frase, che fornisce le stesse informazioni delle relazioni
"Una **persona** risiede in un **paese**" e "Il **paese** si trova in una **provincia**".

È quindi eliminabile la relazione tra Persona e Provincia, riducendo il diagramma E-R a:

Vediamo un secondo esempio.

ESEMPIO

Consideriamo la seguente situazione:
"Ogni **città** appartiene a una **nazione** e a un **continente**, ogni **continente** è formato da **nazioni**."

1) Realizzazione del diagramma E-R

Le tre entità Città, Continente e Nazione hanno tra loro le seguenti relazioni:

2) Definizione delle chiavi

In questo esempio aggiungiamo anche la definizione delle chiavi: sappiamo che le chiavi artificiali iniziano con ID. Come già detto, per convenzione di rappresentazione usiamo ID maiuscolo per la chiave primaria mentre indichiamo con id minuscolo le chiavi esterne.

3) Affinamento

La relazione tra Città e Continente è ridondante e può essere eliminata: come nell'esempio precedente, questa relazione è una duplicazione di informazioni.

Infatti la chiave esterna Id_Continente(fk) nell'entità Città ha una funzione informativa uguale alla "catena" delle chiavi esterne Id_Nazione(fk) + Id_Continente(fk) presenti rispettivamente in Città e Nazione.

Lo schema relazionale è quindi il seguente:

Città (ID_Città (pk), Id_Nazione (fk))
Nazione (ID_Nazione (pk), Id_Continente (fk))
Continente (ID_Continente (pk))

Un esempio completo: gestione di dati di un archivio fotografico

SITUAZIONE

Si vuole realizzare la base di dati di un archivio fotografico di una agenzia avente più sedi, per catalogare le fotografie in base a un catalogo di soggetti definiti in un elenco.
Le foto hanno una dimensione e uno stato di conservazione; per le foto a colori, è noto il tipo di stampa (chiaro o opaco).
Per ogni sede è definito un responsabile, l'indirizzo, il numero telefonico e l'orario di apertura.
Le foto possono descrivere personaggi o luoghi:

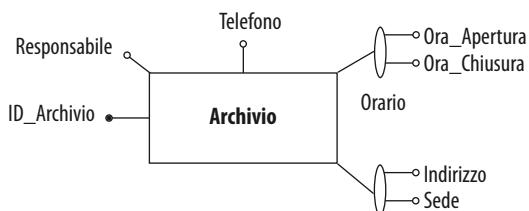
- **i personaggi hanno un nome e un sesso, data di nascita e, eventualmente, di morte, e possono essere:**
  - **artisti, nel qual caso si indica la loro attività prevalente (pittura, scultura, ...);**
  - **sportivi, nel qual caso si indica il loro sport e la squadra di appartenenza;**
  - **uomini politici, nel qual caso si indica il partito e il periodo di appartenenza a esso;**
- **quando le foto descrivono luoghi, è noto il nome della città e la descrizione.**

SOLUZIONE

Per risolvere questa situazione applichiamo lo schema operativo che prevede la iniziale costruzione del glossario, per poi procedere alla progettazione concettuale e, quindi alla progettazione logica del nostro database.

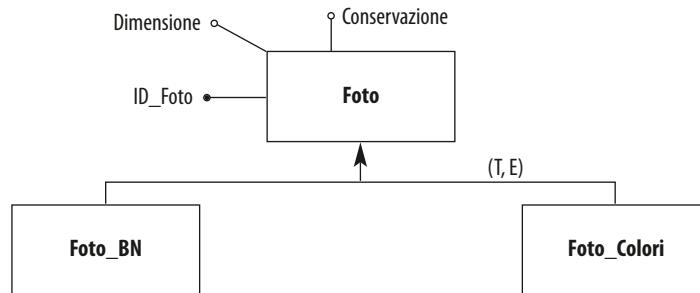
Costruzione del glossario

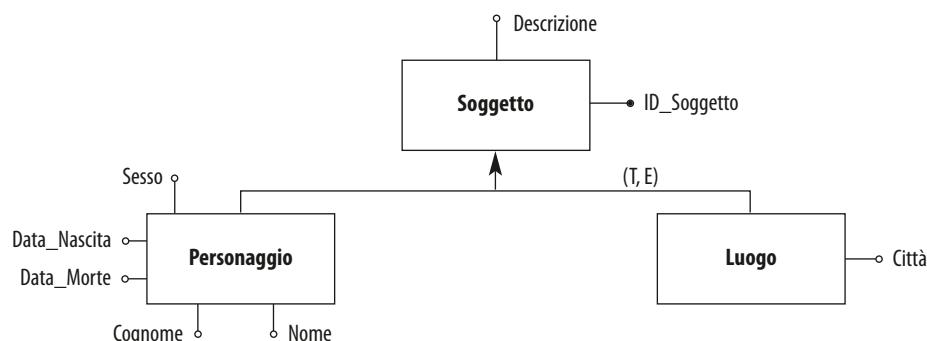
Dopo una prima lettura del testo individuiamo i termini del glossario.

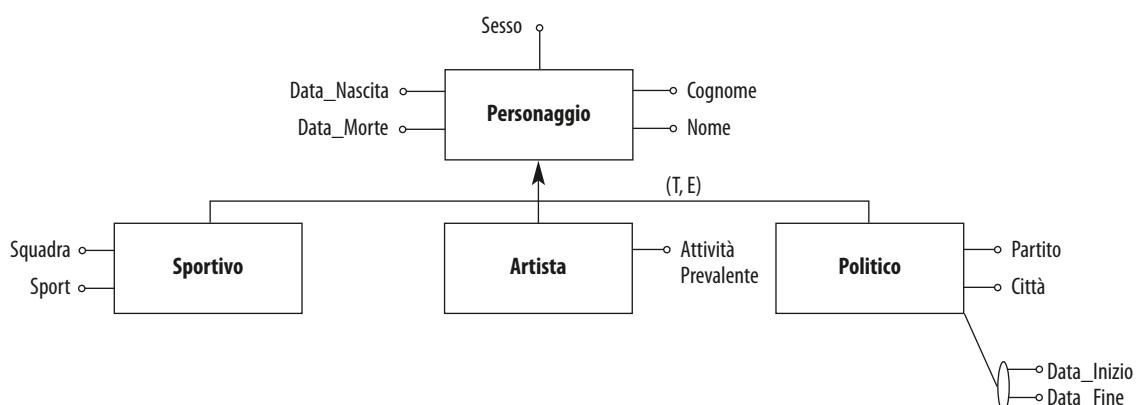
TERMINE	DESCRIZIONE	SINONIMO	LEGAME
Archivio	Raccoglitore di più fotografie a soggetto Esiste un referente	Schedario	Sede Fotografia
Sede	Luogo ove è fisicamente posizionato l'archivio	Città	Archivio Responsabile
Fotografia	Sono di diversi formati A colori oppure in BN Di due tipologie: personaggi o luoghi		Soggetto
Soggetto	Personaggio o luogo		Artista, Atleta, Politico, Luogo
Attività artisti	Pittura, scultura ecc.		Artista
Squadra atleti	Società sportiva che fa effettuare uno (o più) sport agli atleti	Società sportiva	Atleta, Sport
Politico	Viene ritratto in una foto		Fotografia
Città	Ove risiede l'archivio Luogo immortalato nella foto	Paese	Luogo, Sede, Fotografia

Progettazione concettuale

Iniziamo quindi a individuare le entità fondamentali, che sono Archivio, Fotografie e Soggetto, e le relazioni che tra loro intercorrono per poter costruire il diagramma E-R.

Entità Archivio

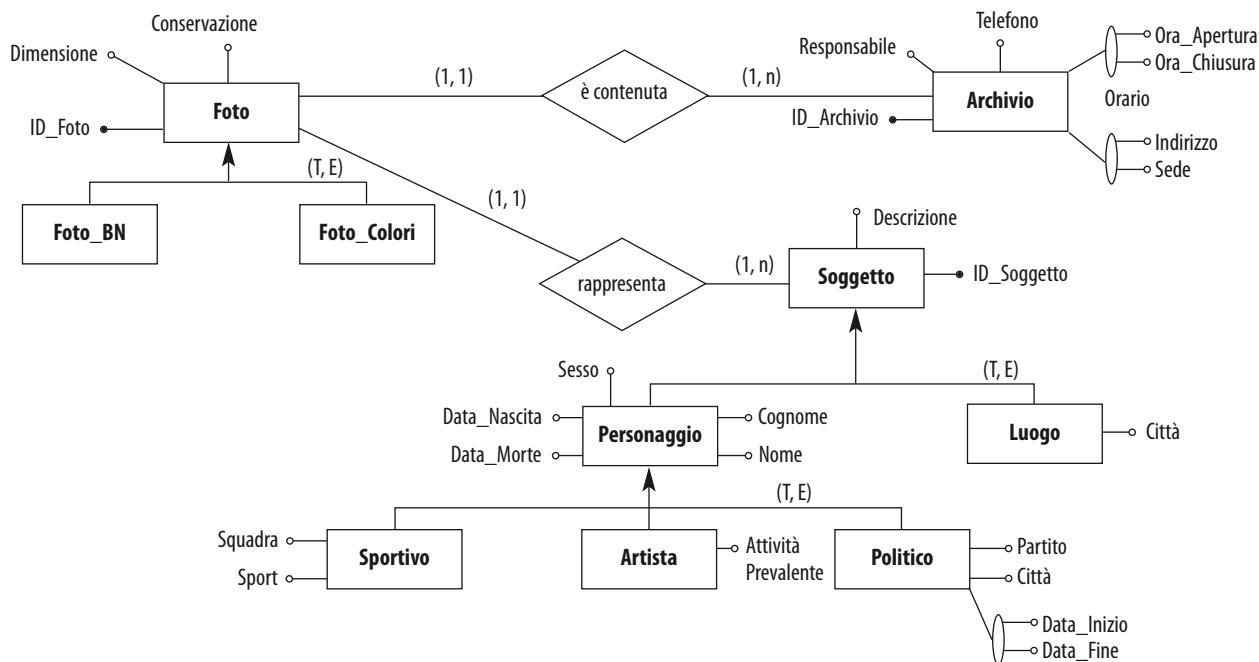
Direttamente dal testo si possono individuare gli attributi e le caratteristiche di questa entità, ipotizzando che l'orario di apertura sia continuato e quindi siano necessari due attributi, Ora_apertura e Ora_chiusura, per definirlo.

Introduciamo una chiave artificiale come chiave primaria.

Per semplicità, indichiamo il campo Indirizzo come campo atomico, come viene generalmente considerato, senza scomporlo nelle sue tre componenti.

Entità Fotografie

Le fotografie, come indicato dal testo, possono essere di due tipologie: bianco e nero e a colori. Questa scomposizione permette di individuare una gerarchia totale (T), dato che tutte le fotografie sono di questi due tipi, ed esclusiva (E), dato che o sono a colori o sono in bianco e nero. Introduciamo quindi una chiave artificiale come chiave primaria e aggiungiamo i due attributi indicati dal testo. Le foto inoltre rappresentano un Soggetto, che analizziamo come entità, vista la sua complessità.

Entità Soggetto

Il soggetto può essere un luogo oppure un personaggio: introduciamo quindi una gerarchia totale (T), dato che tutti i soggetti sono di queste due tipologie, ed esclusiva (E), dato che ipotizziamo che o riguardano un Luogo oppure un Personaggio.

Introduciamo una chiave artificiale come chiave primaria e un attributo Descrizione, comune a entrambe le entità-figlie: a esse aggiungiamo invece gli attributi specifici, cioè il Nome, il Cognome, il Sesso e la Data_Nascita - Data_Morte per il Personaggio, la Città per il Luogo.

Inoltre, dal momento che l'entità Personaggio si scompone in tre sottocategorie, introduciamo un'ulteriore gerarchia.

Relazioni tra entità

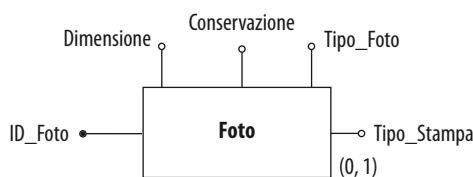
Tra queste tre principali entità individuiamo le seguenti due relazioni:
– la Fotografia è contenuta nell'Archivio;
– la Fotografia rappresenta un Soggetto.

A questo punto, possiamo disegnare lo schema E-R completo.

METTITI ALLA PROVA

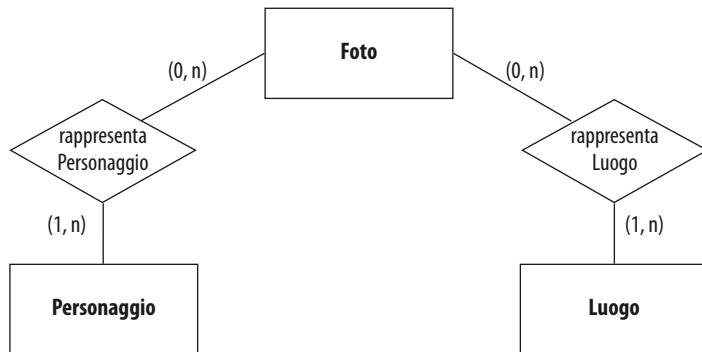
➜ Progettazione E-R • Riduzione dello schema

Completa l'esempio introducendo come soggetto una nuova entità, Opera d'arte, della quale si vuole memorizzare l'artista che l'ha realizzata e il luogo dove è conservata.

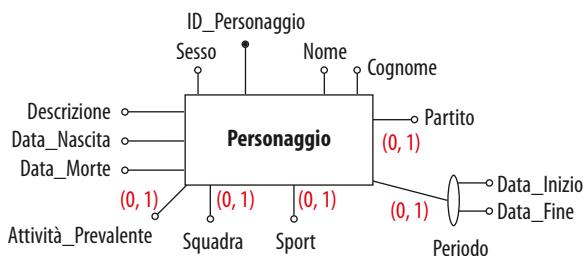
Progettazione logica

Procediamo ora all'analisi dello schema E-R ottenuto e alla sua ristrutturazione.

1. Eliminazione gerarchie

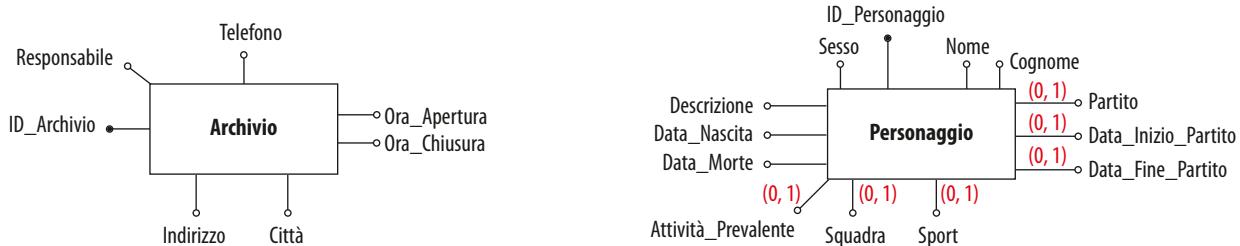
Eliminiamo le gerarchie presenti nelle entità Foto e Soggetto.

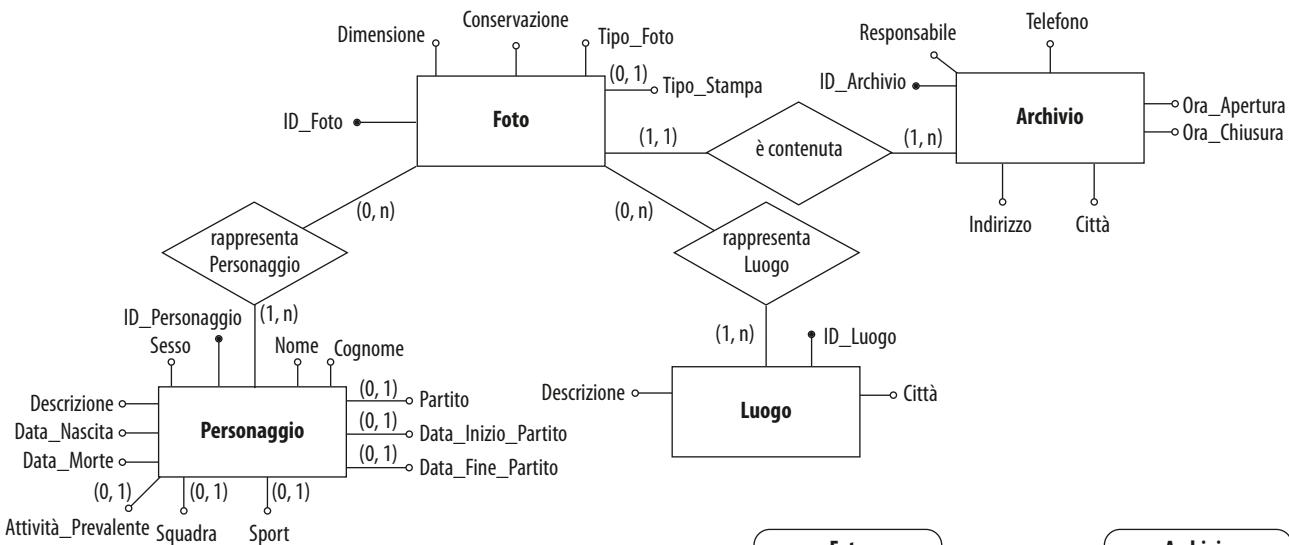
Foto

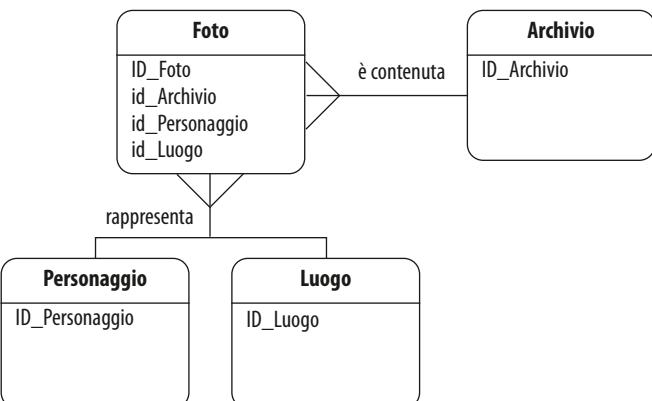
La semplice distinzione tra Foto_BN e Foto_Colori ci autorizza a far collassare le entità figli nel padre Foto aggiungendo un attributo obbligatorio Tipo_Foto con dominio (BiancoNero, Colori) e un attributo opzionale, richiesto solo nel caso di foto a colori, per definire il Tipo_Stampa.
Il diagramma E-R definitivo è il seguente.

Soggetto

L'entità Soggetto ha una gerarchia a tre livelli ed è preferibile collassare verso il livello intermedio, dato che la gerarchia è totale ed esclusiva, mantenendo quindi due entità e modificando la relazione mediante la sua scomposizione in due relazioni.

Nella entità Personaggio vengono aggiunti gli attributi necessari a dettagliare le tre possibili alternative, indicandoli come opzionali dato che saranno inseriti i dati solo nelle specifiche situazioni indicate dal testo.

Nella entità Luogo non è necessario alcun affinamento.

2. Attributi multipli

Per quanto riguarda gli attributi sono presenti due situazioni da ristrutturare nella entità Archivio e una nella entità Personaggio: sono tutte situazioni di attributi multipli.

- Orario: aggiungiamo direttamente i due attributi Ora_Apertura e Ora_Chiusura nell'entità Archivio.
- Sede: aggiungiamo direttamente gli attributi Indirizzo e Città nell'entità Archivio.
- Periodo: nell'entità Personaggio modifichiamo i nomi degli attributi Data_Inizio e Data_Fine in Data_Inizio_Partito e Data_Fine_Partito e li aggiungiamo come opzionali all'entità aggiungendo anche l'attributo che indica il Partito di appartenenza.

3. Analisi delle chiavi

Analizzando tutte le chiavi possiamo osservare che è necessaria una chiave artificiale per l'entità Luogo così come è stata introdotta una chiave artificiale per l'entità Personaggio: aggiungiamo quindi ID_Luogo come chiave primaria.

Lo schema E-R finale è il seguente:

La schematizzazione sintetica mediante il diagramma UML (a lato) è invece:

Da questo esempio è possibile "riflettere" sulla diversità, qualità e quantità di dettagli offerti dalle due diverse grafiche di rappresentazione del modello E-R.

4. Schema relazionale

Dallo schema E-R finale ottenuto dalla ristrutturazione possiamo ricavare lo schema relazionale semplice delle singole tabelle:

Luoghi (ID_Luogo(pk), Città, Descrizione)
Personaggi (ID_Personaggio(pk), Cognome, Nome, Sesso, Descrizione, Data_Nascita, Data_Morte, Partito, Data_Inizio_Partito, Data_Fine_Partito, Attività_Prevalente, Squadra, Sport)
Foto (ID_Foto(pk), Dimensione, Conservazione, Tipo_Foto, Tipo_ Stampa, id_ Archivio(fk), id_Personaggio(fk), id_Luogo(fk))
Archivi (ID_Archivio(pk), Responsabile, Indirizzo, Città, Telefono, Ora_Apertura, Ora_Chiusura)

METTITI ALLA PROVA

➜ Ristrutturazione diagramma E-R • Schema relazionale

Partendo dallo schema E-R che comprende l'entità Opera d'arte procedi con la sua ristrutturazione e con la definizione dello schema relazionale. Descrivi le tabelle dello schema relazionale mediante uno schema completo, riportando cioè per ogni attributo i seguenti dettagli.

TABELLA LUOGHI				
Attributo/campo	Tipo	Dimensione	Valori	Note
ID_luogo	Numerico	8	autoincrement	pk
Città	Carattere	40	Non nullo	
…				

VERIFICA... le competenze

PROGETTAZIONE DIAGRAMMA E-R E SCHEMA RELAZIONALE

AREA DIGITALE

Esercizi per il recupero e il rinforzo

Date le seguenti situazioni, definisci il diagramma E-R, quindi ristrutturalo, indicando le eventuali ipotesi aggiuntive, per arrivare alla definizione dello schema relazionale.

1 Automobili e proprietari

Si vuole progettare una base di dati per gestire le informazioni di automobili, descritte da una targa, un modello, una marca e un colore e dei rispettivi proprietari, che possono anche essere più di uno, identificati da un codice fiscale, nome e cognome, sia come proprietari attuali che come eventuali proprietari passati, nel qual caso è anche necessario indicare il periodo di possesso dell'autovettura. Ricordiamo che un proprietario può possedere più automobili.

2 Stati confinanti

Si vuole progettare una base di dati per gestire le informazioni relative agli Stati (nome, continente, numero di abitanti, superficie e densità) , a quelli a essi confinanti e ai rispettivi presidenti, nel caso di repubblica, di cui si conosce il nome, il cognome e l'eventuale partito politico di appartenenza e la data di elezione, oppure a un re (o una regina), nel caso di monarchia, del quale si conosce il nome, il cognome e il casato di appartenenza.

3 Film

Si vuole progettare una base di dati per gestire le informazioni su film e proiezioni: per le proiezioni è richiesto di memorizzare la città, la sala, la data, l'ora, il numero di spettatori per proiezione mentre per i film, che sono identificati da un codice, interessa il titolo, l'anno di produzione, il regista, gli attori che vi recitano e, per ogni attore, se è protagonista o non protagonista e la sua nazionalità.

4 Ufficio IMU

Un Comune richiede la realizzazione di un database per la gestione dell'IMU: vanno indicati i dati delle abitazioni e dei rispettivi proprietari. Per ogni abitazione, identificata da un codice, interessa il tipo (appartamento, villa, ...), la superficie, il numero di vani, eventuali annessi (garage, cantina, giardino, ...), la rendita catastale, il tipo di utilizzo (abitazione principale, libera, in affitto), mentre per ogni proprietario, identificato dal codice fiscale, interessa cognome, nome, indirizzo e, per ciascuna abitazione di cui sono proprietari, la quota di possesso.

5 Ufficio TARI (ex TARSU)

L'ufficio TARI vuole integrare il database comunale già presente per l'IMU aggiungendo la tassa rifiuti che deve essere pagata da chi occupa l'appartamento, che sia proprietario oppure inquilino. Per le abitazioni in affitto occorre memorizzare gli estremi del contratto di locazione, conduttore/i e il locatore/i (codice fiscale, cognome, nome, luogo e data di nascita) e il canone annuo di locazione.

6 Prenotazioni alberghiere

Si vuole progettare una base di dati per gestire le prenotazioni di un albergo, che ha disposizione di diverse tipologie di camere (singole, matrimoniali, suite ecc.): la prenotazione può essere fatta da un cliente per un dato periodo di giorni (data di arrivo e di partenza) indicando il tipo della camera desiderata e, nel caso che in passato sia già stato cliente della struttura, aggiungendo altre preferenze (come numero di camera, esposizione, piano). Per i nuovi clienti è necessario memorizzare i dati anagrafici (il nome, l'indirizzo e il telefono) e i dati fiscali (codice fiscale, data a luogo di nascita, tipo e dati del documento di riconoscimento).

7 Disponibilità alberghiera

Si vuole progettare la base di dati per gestire le disponibilità alberghiere di una stazione turistica dove gli alberghi sono identificati dal nome: di un albergo interessano telefono, fax, indirizzo e categoria, se è a condu-

VERIFICA... le competenze

zione familiare o di proprietà di una multinazionale e se accetta animali (e, in caso affermativo, quali tipologie). Giornalmente ogni struttura comunica la disponibilità in termini di tipologia della camera (singola, doppia, o matrimoniale) per i successivi 30 gg.

8 Autonoleggio

Si vuole progettare una base di dati per una ditta di autonoleggio; le informazioni da archiviare riguardano le automobili, le tariffe, i clienti. Per le automobili interessa la targa, la marca, il modello, l'anno di immatricolazione, la classe tariffaria di appartenenza; una classe tariffaria è identificata da un codice ed è costituita da una quota fissa giornaliera, una quota per km percorso e una quota assicurativa. Per i clienti si vuole memorizzare il codice fiscale, il nome, vari numeri di telefono, se ha effettuato incidenti nei precedenti noleggi e, in tale caso, avrà un aumento in % sulla tariffa in base a una tabella Malus (per esempio, con 1 incidente 3% di aumento, con 2 incidenti 5% ecc.).
Le auto della agenzia possono essere disponibili oppure in noleggio: delle auto in noleggio, oltre al cliente che l'ha noleggiata, si vuole conoscere la data di inizio del noleggio e i km iniziali.

9 Pubblicazioni scientifiche

Si vuole progettare una base di dati per una azienda ospedaliera che deve gestire le carriere dei propri ricercatori: un ruolo importante viene svolto dalle pubblicazioni scientifiche che ogni ricercatore presenta, che verranno gestite in un archivio contenente per ciascuna di esse il titolo, gli autori, la data di pubblicazione e la categoria medica di appartenenza (es. geriatria, oculistica ecc.).
Una pubblicazione può essere divulgata solo in una e una sola delle seguenti situazioni:
- libro, di cui si conosce l'editore;
- articolo su rivista della quale si conosce il nome;
- convegno, di cui si conosce l'anno e la città.

Nella pubblicazione possono essere presenti riferimenti bibliografici a altre pubblicazioni sia degli stessi autori che di altri.

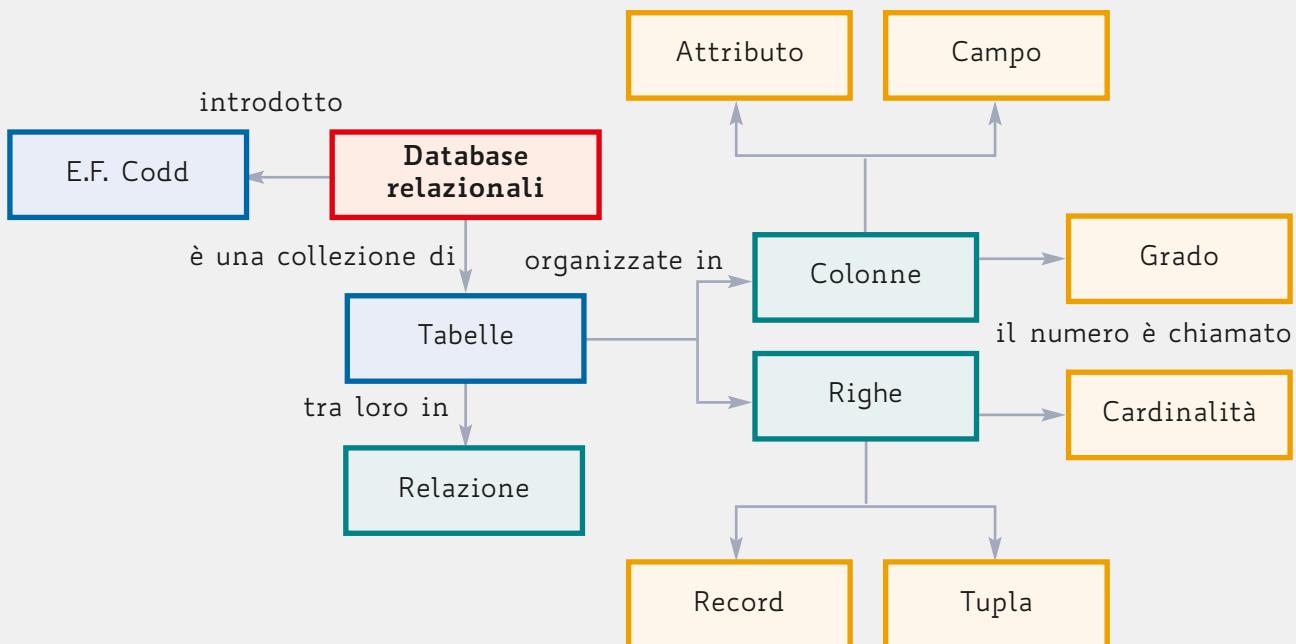
10 Gestione catalogo ristoranti

Una agenzia turistica deve catalogare i ristoranti in base a specifiche caratteristiche, a partire dalle diverse tipologie (lusso, tradizionali, trattorie, paninoteche, pizzerie ecc.) e in base ai diversi tipi di cucina (bolognese, napoletana, cinese, thailandese, italiana ecc.). I ristoranti sono siti in zone diverse della città, ognuna raggiunta da almeno una linea urbana (autobus o metro). I ristoranti, per alcuni dei quali è necessaria la prenotazione, possono anche avere parcheggio riservato e accettare carte di credito o ticket pranzo di enti con cui sono convenzionati.

11 Gestione videoteca casalinga

Si vuole progettare una base di dati per gestire informazioni su una videoteca casalinga, dove i film vengono archiviati per genere e anno di pubblicazione e per i quali si memorizzano gli attori principali e il regista. Di questi si ha a disposizione una tabella con nome, cognome (o lo pseudonimo), l'anno di nascita e la nazionalità. Per ciascun film si vuole memorizzare una breve recensione e un giudizio di valutazione in base a una scala da 1 a 10, il numero di volte che è stato visto e la data dell'ultima visione.

12 Museo

Si vuole progettare una base di dati per gestire informazioni relative a un museo che contiene diverse tipologie di opere, quali dipinti, sculture (divise in statue e bassorilievi), arazzi e ceramiche, che vengono esposte in sale tematiche nelle diverse sezioni nelle quali il museo è organizzato. Ogni sezione è custodita da un solo custode al giorno che a rotazione, settimana per settimana, passa da una sezione a un'altra, mantenendo comunque sempre gli stessi orari di servizio. Per ogni opera deve essere indicato il periodo storico di appartenenza (per esempio Rinascimento, Medioevo ecc.) e l'autore (ove noto) indicando la nazionalità e il movimento artistico di appartenenza.

9 Dallo schema logico alle tabelle del DBMS relazionale

IN QUESTA LEZIONE IMPAREREMO...

- a conoscere la struttura dei dati e la terminologia del database
- quali sono le proprietà delle tabelle relazionali

MAPPA CONCETTUALE

Modello relazionale e database relazionale

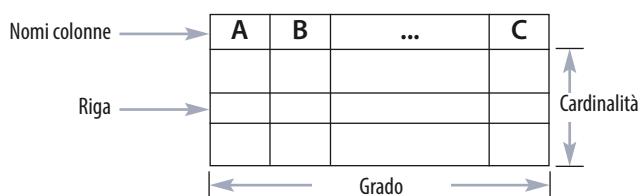
Il modello relazionale, introdotto formalmente da E.F. Codd nel 1970, fornisce una descrizione semplice ma definita rigorosamente di come gli utenti possono vedere i dati all'interno di un database. Il modello relazionale rappresenta i dati nella forma di tabelle bidimensionali, dove ogni tabella rappresenta "qualcosa" di esistente nel mondo reale su cui vengono raccolte informazioni: una persona, un luogo, una cosa o un evento.

Un database relazionale è quindi una collezione di tabelle bidimensionali:
- l'organizzazione dei dati nelle tabelle relazionali è nota come vista logica del database ed è la forma in cui un database relazionale presenta i dati all'utente e al programmatore;
- il modo in cui il software del database salva fisicamente i dati sul disco fisso di un computer è detto vista interna.

La vista interna è diversa da prodotto a prodotto: non sarà quindi trattata in questa unità ma nell'unità 2, dove ci occuperemo dell'implementazione fisica in Access oppure in MySQL.

Sintetizziamo di seguito caratteristiche e peculiarità dei database relazionali:
- fanno uso di tabelle e campi per memorizzare i dati;
- le tabelle hanno uno schema fisso: un nome, un elenco di campi di cui vengono specificati i rispettivi tipi (stringhe di caratteri, numeri, date e dati binari) e una chiave primaria che identifica univocamente una riga della tabella;
- tra due o più campi di tabelle può esserci una relazione, ossia una condizione che lega tra loro le rispettive righe a cui i campi appartengono: le due tabelle sono collegate tramite chiavi esterne e la validità del legame è garantito dai vincoli di integrità referenziale;
- l'accesso ai dati (transazione) viene garantito con le proprietà ACID (Atomicità, Consistenza, Isolamento e Durabilità):

Rollback
È la procedura che ripristina la situazione precedente ad una transazione che non è andata a buon fine, riportando il database nello stato di partenza, precedente l'inizio della transazione stessa.

- Atomicità: ogni transazione non può essere eseguita parzialmente, ma o solo totalmente con successo (commit) o deve fallire totalmente senza modificare il database (**rollback**).
- Consistenza: alla fine della transazione il database si trova in uno stato consistente, ossia tutti i vincoli dello schema del database sono rispettati, altrimenti la transazione avrebbe fallito: interrogando il database dopo la transazione è possibile constatare che le modifiche sono entrate e sono consistenti.
- Isolamento: eseguendo due transazioni in contemporanea, l'effetto finale è uguale a quello che si ottiene eseguendo le transazioni una di seguito all'altra, ossia l'effetto temporaneo di una transazione non ancora completata non è visibile alle altre transazioni.
- Durabilità: una volta che una transazione è terminata con successo, le modifiche sono diventate persistenti anche in caso di crash del sistema subito dopo la transazione.

Una conoscenza di base del modello relazionale è necessaria per usare in modo efficace il software dei database relazionali come Oracle, Ms SQLServer, MySQL o anche altri database minori come Access, che sono basati su tale modello.

Struttura dei dati e terminologia

Abbiamo detto che, nel modello relazionale, un database è una collezione di tabelle. Una tabella relazionale può essere immaginata come un file in cui vengono salvati i dati sotto forma di colonne, a cui viene assegnato un nome e un numero arbitrario di righe. Le colonne contengono informazioni sulla tabella e le righe riportano le istanze degli oggetti rappresentati dalla tabella.

Un singolo dato è salvato nell'intersezione di una riga con una colonna.

Lo schema seguente chiarisce che cosa si intende per tabella nel contesto dei database relazionali.

- L'insieme dei valori che possono essere presenti in una colonna è detto dominio.
- Il numero delle colonne è detto grado.
- Il numero delle righe indica la cardinalità della tabella.

Terminologia

Per definire gli oggetti che appartengono al modello relazionale la matematica ricorre ai termini tabella, riga e colonna, che saranno quelli che verranno utilizzati anche in questa unità di apprendimento.
Per una serie di ragioni storiche, tuttavia, i progettisti di database utilizzano anche, in riferimento agli oggetti che fanno parte di un database relazionale, i termini campo e attributo per indicare le colonne e record e tupla per indicare le righe.

IN QUESTA UNITÀ DI APPRENDIMENTO	NOMI ALTERNATIVI
Tabella	Tabella relazionale / Relazione
Riga	Record / Tupla
Colonna	Attributo / Campo

Funzioni fondamentali dei database relazionali

Finora sono stati descritti gli oggetti principali che fanno parte di un database relazionale.
Definiamo ora il database relazionale tenendo conto dei suoi obiettivi funzionali.

Un database relazionale è un oggetto che:
- permette di realizzare l'archiviazione fisica su memoria di massa dei dati secondo gli schemi concettuali definiti nel modello E-R;
- presenta all'utente/programmatore i dati organizzati secondo la struttura logica di tabelle e di associazioni tra tabelle.

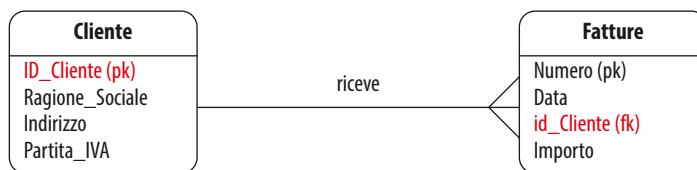
Quindi, il database relazionale è una "evoluzione" del modello E-R descritto nelle lezioni precedenti e le tabelle definite nello schema logico saranno la base per la realizzazione fisica del database.

Le fasi di lavoro per la definizione e la realizzazione di un database relazionale si possono sintetizzare come riportato nella tabella che segue:

LIVELLO	FASE	A CURA DI
Concettuale	Definizione del modello E-R	Utente finale Progettista DB
Logico	Definizione delle strutture dei dati	Amministratore del DB
Fisico	Archiviazione e gestione delle strutture dei dati	DBMS

L'amministratore del database (DBA, *Data Base Administrator*), citato nella tabella sopra riportata, è una figura informatica che ha la funzione di interfacciarsi con l'utente finale per la definizione del modello E-R e con i programmatori che utilizzano il DBMS.

Dallo schema concettuale allo schema logico

Nella lezione precedente abbiamo visto come definire la struttura logica dei dati all'interno di un database partendo dal modello E-R ristrutturato: sintetizziamo il procedimento in queste quattro semplici regole "meccaniche":
1 ogni entità diventa una tabella;
2 un'istanza di un'entità diventa una riga della tabella;

3 ogni attributo dell'entità diventa una colonna della tabella;
4 la chiave primaria dell'entità è l'identificatore univoco delle righe di una tabella.

Vediamo un esempio partendo da un semplice schema E-R finale composto da due entità.

ESEMPIO

Tra le due entità Cliente e Fattura lo schema concettuale finale è il seguente:

La trasformazione nello schema logico è immediata: abbiamo due tabelle, una per ciascuna entità, alle quali, per convenzione, daremo il nome plurale: Clienti e Fatture.
Possiamo quindi scrivere il seguente schema relazionale:

Clienti (ID_Cliente(pk), Ragione_Sociale, Indirizzo, Partita_IVA)
Fatture (Numero(pk), Data, Importo, id_Cliente(fk))

Gli attributi diventano colonne, e le chiavi sono generalmente indicate nelle colonne estreme:
- la chiave primaria nella/e prima/e colonna/e di sinistra;
- la/e chiave/i esterna/e nelle colonne di destra.

Le tabelle così ottenute costituiscono lo schema logico del database: popoliamo le due tabelle inserendo alcune istanze.

Tabella Clienti

ID_Cliente	Ragione_Sociale	Indirizzo	Partita_IVA
120	Rossi SRL	Via Roma, 3	10203040501
220	Verdi snc	Via Torino, 23	20304040501
333	Gialli SPA	Via Napoli, 11	40503040501
520	Rossi SRL	Via Genova, 13	30503040501

Tabella Fatture

Numero	Data	Importo	id__cliente
70	01/04/2017	1200,20	120
71	03/04/2017	1450,29	520
72	04/04/2017	1200,20	120
73	05/04/2017	2300,45	220
74	06/04/2017	1200,20	120

È vivamente consigliato popolare le tabelle, in modo da verificare che il DB ottenuto rispecchi le specifiche di progetto e, allo stesso tempo, per predisporre i dati per le successive operazioni di normalizzazione e collaudo.

Proprietà delle tabelle relazionali

Le tabelle relazionali hanno le seguenti proprietà:
- i valori sono atomici;
- i valori di una colonna sono dello stesso tipo, cioè appartengono allo stesso dominio;
- ogni riga è univoca;
- la sequenza delle colonne non è significativa;
- la sequenza delle righe non è significativa;
- ogni colonna deve avere un nome univoco.

Esaminiamo di seguito queste proprietà più nel dettaglio.

I valori sono atomici

Questa proprietà implica che le colonne nelle tabelle relazionali non sono ulteriormente scomponibili e non sono organizzate in vettori.

ESEMPIO

Il nome e il cognome di un autore nella tabella Autori non possono essere raggruppati in unica colonna denominata Nome.

AUTORI				
ID_Autore	**Nome**	**Indirizzo**	**Città**	**CAP**
172	Bianchi Mario	Via Garibaldi, 10	Brescia	25100
156	Cerreti Giovanni	Via Carducci, 12	Milano	20100
268	Rossigni Giacomo	Piazza Caduti, 20	Torino	10100
884	Tacchi Alfio	Borgo Trento	Treviso	31100
343	Bertozzi Filippo	Via del lago, 9	Arenzano	16011
298	Sommo Norberto	Largo Roma, 3	Como	22100

Per essere correttamente strutturata, la tabella deve avere due colonne, rispettivamente per il nome e per il cognome, come riportato di seguito.

AUTORI					
ID_Autore	**Cognome**	**Nome**	**Indirizzo**	**Città**	**CAP**
172	Bianchi	Mario	Via Garibaldi, 10	Brescia	25100
156	Cerreti	Giovanni	Via Carducci, 12	Milano	20100
268	Rossigni	Giacomo	Piazza Caduti, 20	Torino	10100
884	Tacchi	Alfio	Borgo Trento	Treviso	31100
343	Bertozzi	Filippo	Via del lago, 9	Arenzano	16011
298	Sommo	Norberto	Largo Roma, 3	Como	22100

La proprietà di atomicità dei valori nelle tabelle relazionali è di grande importanza perché è stata una delle pietre angolari del modello relazionale. Il maggiore vantaggio della proprietà atomica dei valori è costituito dalla possibilità di adottare logiche di manipolazione dei dati più semplici e più efficienti.

PER SAPERNE DI PIÙ

ATTRIBUTI ATOMICI

Nella tabella dell'esempio precedente si può osservare che anche l'indirizzo non è atomico: dovremmo quindi, per coerenza con quanto sopra affermato, scomporre la colonna Indirizzo in tre colonne, in modo da separare la tipologia, il nome e il numero civico, come nella tabella seguente.

AUTORI							
ID_Autore	**Cognome**	**Nome**	**Tipologia strada**	**Indirizzo**	**Num**	**Città**	**CAP**
172	Bianchi	Mario	Via	Garibaldi	10	Brescia	25100
156	Cerreti	Giovanni	Via	Carducci	12	Milano	20100
268	Rossigni	Giacomo	Piazza	Caduti	20	Torino	10100
298	Sommo	Norberto	Largo	Roma	3	Como	22100

Nella pratica, tuttavia, si preferisce avere un modello più semplice dei dati e quindi tutti gli attributi vengono di fatto suddivisi soltanto quando contengono dati di cui si prevede un effettivo utilizzo nelle successive operazioni, come vedremo in seguito nella fase di normalizzazione delle tabelle.
L'attributo Indirizzo, nel caso in cui nelle specifiche non sia prevista la ricerca dei soggetti che abitano nella stessa via, si lascia quindi in un unico campo, vale a dire in una colonna unica.

Lo stesso discorso vale per il numero telefonico, che molto difficilmente si separa in due campi isolando il prefisso dal numero vero e proprio.

I valori delle colonne sono dello stesso tipo

In termini relazionali, questo significa che tutti i valori in una colonna appartengono allo stesso dominio.

Un dominio è un insieme di valori che possono essere inseriti in una colonna.

ESEMPIO

La colonna Nome della tabella Autori può contenere solo caratteri alfanumerici e non numeri o altri tipi di dati che siano diversi dal nome di un autore.

Questa proprietà semplifica l'accesso ai dati, poiché gli sviluppatori e gli utenti possono essere certi del tipo di dati che sono contenuti in una data colonna; semplifica inoltre anche il processo di validazione dei dati stessi.
Poiché tutti i valori appartengono allo stesso dominio, quest'ultimo può essere definito e rafforzato con il *Data Definition Language* (DDL) del software di gestione del database.

Ogni riga è univoca

Questa proprietà assicura che, in una tabella relazionale, non ci siano due righe identiche; deve quindi esserci almeno una colonna o un insieme di colonne i cui valori identificano univocamente ogni riga nella tabella.

Queste colonne, come già sappiamo, sono chiamate chiavi primarie.

Questa proprietà garantisce che, in una tabella relazionale, ogni riga abbia significato e che una riga specifica possa essere identificata attraverso il valore della sua chiave primaria.

La sequenza delle colonne non è significativa

Questa proprietà indica che, in una tabella relazionale, l'ordine delle colonne non ha alcun significato.

Le colonne possono essere lette in qualsiasi ordine e in varie sequenze.

Il beneficio offerto da questa proprietà è che permette a molti utenti di condividere la stessa tabella senza preoccuparsi di come la tabella stessa sia organizzata; questo consente inoltre di cambiare la struttura fisica del database senza effetti/modifiche sulle tabelle relazionali.

È dunque possibile aggiungere colonne anche in tempi successivi alla creazione del database senza dover modificare i programmi esistenti.

La sequenza delle righe non è significativa

Questa proprietà è analoga alla precedente, ma si applica alle righe invece che alle colonne.

Il maggior beneficio consiste nel fatto che le righe di una tabella relazionale possono essere recuperate in ordini e sequenze diversi.

L'aggiunta di informazioni a una tabella relazionale avviene in modo semplificato e non influenza le interrogazioni in corso: questo significa che i record nell'archivio sono svincolati dalla loro posizione logica e quindi il programmatore non deve mai fare riferimento a tale valore per individuare un elemento.

ESEMPIO

Un dato che attualmente si trova alla riga numero 27 potrebbe, nel corso della vita del database, cambiare molteplici posizioni: se viene cancellato un record che lo precede, esso "shifta" automaticamente indietro di una posizione, assumendo la posizione 26, e via di seguito.

Ogni colonna deve avere un nome univoco

Poiché la sequenza di colonne non è significativa, si deve fare riferimento alle colonne per nome e non per posizione.
In generale, il nome di una colonna non ha bisogno di essere univoco in tutto il database, ma solo all'interno della tabella di cui fa parte.

Un esempio completo: database bibliografico

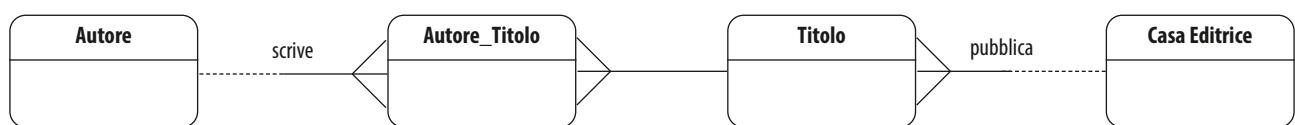
SITUAZIONE

Si vuole realizzare un database bibliografico contenente informazioni sui volumi pubblicati con la registrazione dei titoli dei libri, degli autori e delle case editrici.

SOLUZIONE

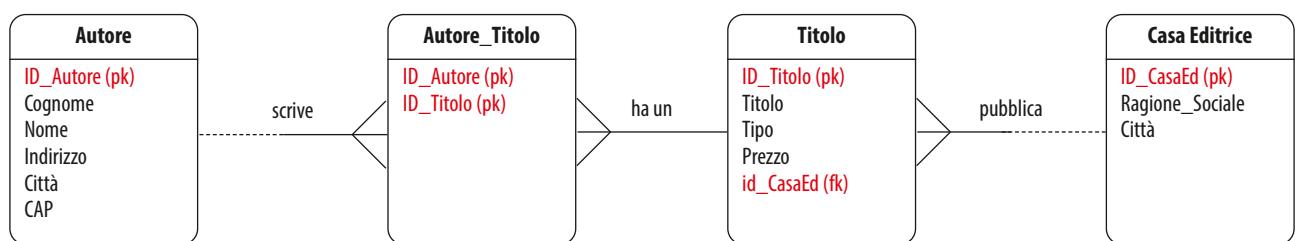
Vediamo ora i passaggi che, dal modello E-R, ci consentono di giungere al database relazionale.

Analisi dei dati

Si possono individuare, come primo passo, le entità principali, che sono:

Schema scheletrico

Come secondo passo individuiamo le relazioni presenti tra di esse:

Affinamento diagramma E-R

Poiché tra Autore e Titolo è presente una relazione molti-a-molti, si può procedere a un affinamento del modello E-R, introducendo tra di esse un'entità associativa:

Completamento dello schema E-R con gli attributi

A ciascuna entità associamo adesso gli attributi essenziali con l'indicazione delle chiavi primarie ed esterne.

In un caso reale, gli attributi delle varie entità dovrebbero essere molto più numerosi: qui ci limitiamo ai più ovvi e soltanto a quelli per non appesantire la trattazione.

Definizione schema logico e relazionale

Dal diagramma E-R ristrutturato ricaviamo il seguente schema relazionale:

Autori (ID_Autore(pk), Cognome, Nome, Indirizzo, Città, CAP)
Titoli (ID_Titolo(pk), Titolo, Tipo, Prezzo, id_CasaEd(fk))
Case Editrici (ID_CasaEd(pk), Ragione_Sociale, Città)
Autori_Titoli (ID_Autore(pk), ID_Titolo(pk))

Riproduciamo quindi graficamente lo schema logico popolando le tabelle:

AUTORI					
ID_Autore	**Cognome**	**Nome**	**Indirizzo**	**Città**	**CAP**
172	Bianchi	Mario	Via Garibaldi, 10	Brescia	25100
156	Cerreti	Giovanni	Via Carducci, 12	Milano	20100
268	Rossigni	Giacomo	Piazza Caduti, 20	Torino	10100
884	Tacchi	Alfio	Borgo Trento	Treviso	31100
343	Bertozzi	Filippo	Via del lago, 9	Arenzano	16011
298	Sommo	Norberto	Largo Roma, 3	Como	22100

AUTORI_TITOLI	
ID_Autore	**ID_Titolo**
172	7771010
156	7771213
268	7772123
884	7773244
343	7771356
298	7774744

TITOLI				
ID_Titolo	**Titolo**	**Tipo**	**Prezzo**	**id_CasaEd**
7771010	C++	Informatica	20.00	91234
7771213	*La cucina mediterranea*	Cucina	18.50	92233
7772123	*Aumentare gli utili*	Economia	36.70	94354
7773244	*L'ultima neve*	Thriller	15.20	95555
7771356	*Stress: il nostro nemico*	Psicologia	18.40	94354
7774744	*Le leggi di Mendel*	Biologia	21.00	95555

CASE EDITRICI		
ID_CasaEd	**Ragione Sociale**	**Città**
91234	Tecniche OK	Brescia
92233	Nuove Gastronomie	Milano
94354	Hoepli	Milano
94354	Mondo Libri	Torino
95555	New Gothics	Treviso

Seguendo i collegamenti delle chiavi, possiamo recuperare tutti i dati che sono "sparsi" nelle diverse tabelle.

Libro: *Stress: il nostro nemico*
Argomento: Psicologia
Prezzo: 18,40 euro
Autore: Bertozzi Filippo, Via del lago, 9 16011 - Arenzano
Editore: Mondo Libri, Torino

Conclusioni: schema logico, fisico e tracciato record

Riassumendo, lo scopo della fase di progettazione logica è quello di produrre uno schema logico applicando le regole e i concetti del modello concettuale che si è adottato, rappresentando le informazioni che sono state descritte nella fase di progettazione concettuale attraverso la realizzazione dello schema E-R.

Il procedimento di conversione che porta, mediante l'analisi delle informazioni riguardanti le entità, gli attributi e le associazioni, dallo schema concettuale allo schema logico, si chiama mapping.

Lo schema logico dei dati definisce la struttura logica dei dati, cioè il modo in cui le informazioni contenute negli archivi vengono aggregate, indipendentemente dagli archivi fisici in cui esse sono contenute.

Partendo da uno schema concettuale, la creazione di uno schema logico relazionale passa attraverso le seguenti regole:

- le entità dello schema concettuale diventano tabelle;
- le relazioni vengono rappresentate nello schema logico relazionale ricorrendo a chiavi esterne, che consentono di stabilire un riferimento tra le righe delle due tabelle.

Il modello relazionale è basato unicamente su valori; la prima parte del progetto dello schema logico consiste quindi nel rappresentare le relazioni o in forma grafica oppure in forma testuale, trasformandolo nelle seguenti relazioni:

Autori (ID_Autore(pk),Cognome, Nome, Indirizzo, Città, Cap)
Titoli (ID_Titolo(pk),Titolo, Tipo, Prezzo, id_CasaEd(fk))
Case Editrici (ID_CasaEd(pk), Nome, Città)
Autori_titoli (ID_Autore(pk), ID_Titolo(pk))

La definizione del modello logico prevede poi la verifica delle regole di integrità e di normalizzazione, che verranno descritte nelle prossime lezioni.
Successivamente alla definizione del modello logico, si procede con la specifica dei parametri che serviranno per la memorizzazione dei dati da parte del DBMS.

Questa fase prende anche il nome di progettazione fisica.

Una formulazione intermedia è il "modello logico preciso", o tracciato record, dove per ogni tabella dello schema logico si indicano le specifiche di ogni campo, cioè il tipo e la dimensione; se si vuole essere ancora più "precisi", si specificano anche i valori che l'attributo può assumere.
Viene spesso completato con una colonna dove si aggiungono eventuali note per evidenziare qualche campo particolare e aggiungere osservazioni utili anche agli sviluppatori delle applicazione che utilizzano tale database.
Vediamo per esempio il tracciato record della tabella Titoli.

TITOLI				
Attributo/campo	**Tipo**	**Dimensione**	**Valore**	**Note**
ID_Titolo	Numerico	8	autoincrement	pk
Titolo	Stringa	40	not null	
Tipo	Stringa	20	da elenco predefinito	
Prezzo	Numerico	8,2	not null	
id_CasaEd	Numerico	6	autoincrement	fk

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 L'acronimo DBA significa:
- **a** Data Base Area
- **b** Data Base Administrator
- **c** Data Base Advanced
- **d** Data Base Access

2 Quali sono le due funzioni fondamentali di database relazionale?
- **a** Realizza l'archiviazione fisica su memoria di massa
- **b** Offre gli schemi concettuali per definire un modello E-R
- **c** Presenta all'utente i dati organizzati secondo la struttura logica
- **d** Definisce dalla struttura E-R la struttura fisica di un archivio

3 Quale tra le seguenti non è una regola per derivare le tabelle?
- **a** Ogni entità diventa una tabella
- **b** Un'istanza di un'entità diventa una riga della tabella
- **c** Ogni attributo dell'entità diventa una colonna della tabella
- **d** Ogni figlia diventa una chiave esterna
- **e** La chiave primaria dell'entità è l'identificatore univoco delle righe di una tabella

4 Quale tra le seguenti non è un proprietà delle tabelle relazionali?
- **a** I valori sono atomici
- **b** I valori di una colonna appartengono al medesimo dominio
- **c** Ogni riga è univoca
- **d** La sequenza delle colonne è significativa
- **e** La sequenza delle righe non è significativa
- **f** Ogni colonna deve avere un nome univoco

5 Quale tra le seguenti affermazioni riferite alle chiavi è errata?
- **a** Una chiave primaria è una colonna o un insieme di colonne
- **b** Una chiave primaria identifica univocamente una riga nella tabella
- **c** Una chiave primaria deve avere valori diversi in ogni tabella
- **d** Una chiave esterna è una colonna o un insieme di colonne
- **e** Una chiave esterna ha valori uguali della chiave primaria di un'altra tabella

6 Il tracciato record:
- **a** per ogni tabella dello schema logico indica le specifiche di ogni campo
- **b** per ogni relazione indica la molteplicità
- **c** per ogni database indica l'elenco dei record
- **d** per ogni attributo indica a che record appartiene

VERO/FALSO

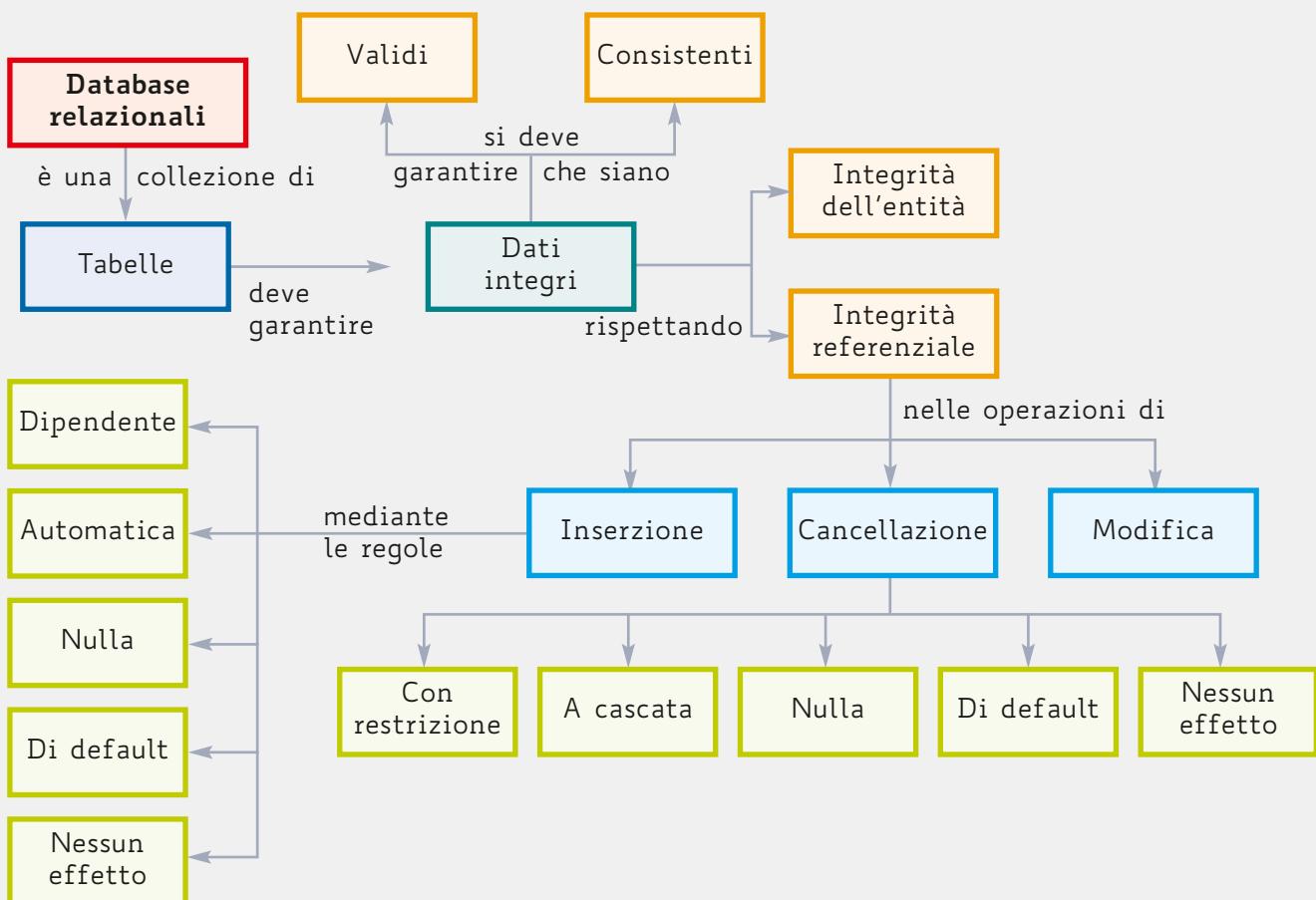
1	Nel modello relazionale, un database è una collezione di tabelle.	V	F
2	Il numero delle colonne è detto grado.	V	F
3	Il numero delle righe indica la cardinalità della tabella.	V	F
4	Il termine tupla è sinonimo di attributo.	V	F
5	Nelle tabelle relazionali la sequenza delle righe non è significativa.	V	F
6	Nelle tabelle relazionali si possono scambiare tra loro le colonne.	V	F
7	La definizione della strutture dei dati è fatta a cura dell'amministratore del DB.	V	F
8	Ogni tabella deve avere una chiave secondaria in relazione con un attributo.	V	F
9	La chiave primaria è una colonna o un insieme di colonne.	V	F
10	La chiave primaria identifica univocamente una riga nella tabella.	V	F
11	La chiave esterna è una colonna dipendente da una tabella principale.	V	F
12	La chiave esterna ha valori uguali alla chiave primaria di un'altra tabella.	V	F
13	Lo schema logico dei dati definisce la struttura logica dei dati.	V	F
14	Il tracciato record indica le specifiche di ogni campo, ovvero il tipo e la dimensione.	V	F

VERIFICA... le competenze

DIAGRAMMA E-R, SCHEMA LOGICO E SCHEMA RELAZIONALE

Per ogni situazione, dopo aver definito il modello E-R indicando le entità, le relazioni, gli attributi e le chiavi, descrivi il progetto mediante lo schema logico relazionale, popolando le tabelle e definendo il tracciato record.

1. **Squadra di pallavolo**: si devono registrare le informazioni sui giocatori, il loro ruolo, le partite effettuate e le squadre avversarie.
2. **Giardino zoologico**: ci sono degli animali appartenenti a diverse specie provenienti da diverse nazioni del mondo; ogni specie è localizzata in un settore avente il nome di un naturalista.
3. **Agenzia di noleggio di autovetture**: ha un parco macchine dove ognuna fa parte di una categoria; per ogni categoria c'è una tariffa di noleggio e devono essere registrati l'autista che la prende in carico e il periodo di noleggio.
4. **Casa discografica**: produce dischi aventi un codice e un titolo; ogni disco è inciso da uno o più cantanti, che possono essere solisti oppure organizzati in un complesso o gruppo musicale.
5. **Reparto ospedaliero**: oltre ai pazienti è necessario memorizzare il medico curante e il periodo di ricovero. È necessario inoltre avere per ciascun paziente i dati di un familiare autorizzato a ricevere le informazioni sullo stato di salute.
6. **Agenzia immobiliare**: ha immobili per la vendita o per la locazione e, oltre ai dati del proprietario, per ogni immobile è necessario memorizzare la tipologia (appartamento, villa, box...), la superficie, il numero di vani, eventuali annessi (garage, cantina, giardino...), il prezzo richiesto e la zona della città in cui si trova.
7. **Antiquario che partecipa a mostre nazionali**: spedisce gli oggetti appartenenti a diverse categorie ed epoche nelle varie città dove tali oggetti vengono messi in vendita. In caso di vendita gli viene comunicato il nome dell'acquirente con i dati per l'emissione della fattura relativa.
8. **Clinica ostetrica**: le pazienti sono ricoverate in camere singole o doppie; per le pazienti che hanno partorito uno o più neonati è necessario trasmettere i dati (nome, data di nascita, sesso ecc.) al Comune di residenza della madre.
9. **Videonoleggio**: sono necessarie le informazioni su attori, registi, film. Di ogni DVD noleggiato è necessario memorizzare il cliente e per quanti giorni è stato trattenuto il film, per poi effettuare una statistica per genere dei film che sono stati più visti nel corso dell'anno.
10. **Archivio fotografico personale**: di ogni foto interessano il nome e la cartella in cui si trova nel computer, la data e il luogo in cui è stata scattata e un codice che la identifica. Sarebbe opportuno anche sapere i nomi delle persone che sono ritratte e l'evento al quale la foto può essere collegata.
11. **Università**: si vogliono raccogliere e organizzare in un database le informazioni sui propri studenti in relazione ai corsi che frequentano e agli esami che sostengono.
12. **Società di servizi**: svolge le pratiche relative a "previdenza", "sanità" e "ufficio elettorale" per conto di alcuni Comuni della Provincia.
13. **Studio medico-sportivo**: si gestisca in particolare l'aspetto sanitario degli atleti che, a seconda della disciplina che praticano, devono effettuare visite di controllo periodiche.

VERIFICA... le conoscenze

14 **Società polisportiva**: gli allenatori organizzano corsi per le diverse discipline solo per i soci tesserati in regola con il versamento della quota sociale.

15 **Catena di negozi**: è costituita da un certo numero di centri di vendita dislocati nelle diverse regioni italiane, dove per ogni centro sono previsti un direttore e alcuni commessi.

16 **Aeroporto**: diverse compagnie aeree offrono voli per le più importanti capitali europee.

17 **Museo**: si vogliono catalogare in un database le opere d'arte presenti. Tali opere sono identificate tramite un codice identificativo, il titolo e il valore commerciale. Il database vuole gestire anche i dati degli artisti, della corrente artistica di appartenenza e della città di nascita.

18 **Distretto di polizia**: si vogliono catalogare i crimini risolti con l'arresto dei criminali in base alla tipologia, alla gravità e, nel caso di crimine organizzato, memorizzare il nome della banda che lo ha commesso e del suo capobanda, oltre a quello di tutti i suoi componenti.

19 **Ditta di autotrasporti**: si devono registrare per ogni viaggio l'autista, l'automezzo che lo effettua e il cliente al quale addebitare il servizio.

20 **Collezione di fumetti**: oltre alla collana e al protagonista principale, si vogliono memorizzare l'autore della singola storia e la casa editrice; se la storia viene suddivisa su più volumi è necessario sapere su quali volumi è presente.

21 **Stabilimento balneare**: si devono memorizzare le prenotazioni di ombrelloni e lettini (disposti su file numerate da 1 a 5, in cui la fila 1 è quella più vicina al mare, e su colonne numerate da 1 a 20), le tariffe per periodo e per tipologia di servizio e i turni del personale di assistenza ai bagnanti.

22 **Dipendenti e ruoli**: si vogliono gestire gli stipendi di ciascun dipendente che fa parte di un solo reparto e collabora a uno o più progetti durante l'anno fiscale per determinare i budget di spesa annui per reparto e per progetto.

23 **Ospedale e ricoveri**: si devono gestire i dati relativi ai pazienti ricoverati nei diversi reparti, agli interventi effettuati e ai medici che li hanno in cura.

24 **Campionato di calcio di serie A**: si deve gestire la classifica punti e marcatori giornata per giornata memorizzando per ogni squadra i risultati e il nome dei giocatori che hanno effettuato goal o subito ammonizioni ed espulsioni.

LEZIONE 10

10 Le regole di integrità

IN QUESTA LEZIONE IMPAREREMO...

- il concetto di integrità dei dati
- a rispettare le regole di integrità
- a utilizzare le gerarchie di generalizzazione
- a perfezionare il modello E-R

MAPPA CONCETTUALE

Database relazionali
è una collezione di
Tabelle
deve garantire
Dati integri
si deve garantire che siano
Validi
Consistenti
rispettando
Integrità dell'entità
Integrità referenziale
nelle operazioni di
Inserzione
Cancellazione
Modifica
mediante le regole
Dipendente
Automatica
Nulla
Di default
Nessun effetto
Con restrizione
A cascata
Nulla
Di default
Nessun effetto

L'integrità dei dati

L'integrità dei dati è una delle colonne portanti del modello relazionale: essa infatti attesta che i valori contenuti nel database sono validi e consistenti.
Quando vengono eseguite operazioni sui dati presenti nel database, come l'inserimento, la modifica o la cancellazione di dati, è necessario che vengano rispettate le relazioni.

ESEMPIO

Supponiamo di avere un archivio per la gestione di una libreria con Libri e Argomenti in relazione tra loro, dove quindi nella tabella Libri viene inserita una chiave esterna che dovrà avere un valore presente nella colonna della chiave primaria di Argomenti. Un libro può trattare un argomento tra quelli registrati e se effettuiamo una modifica, per esempio del codice di un argomento, questa modifica deve contemporaneamente essere effettuata in tutti i record di ciascuna entità che ha questo codice come chiave esterna, in modo da mantenere i collegamenti con tutte le tabelle.
Pertanto, ogni modifica dell'entità Argomenti deve produrre, a cascata, la corrispondente modifica del codice in un'eventuale riga della tabella Libri.

Le regole di integrità che devono sempre essere rispettate sono due:
– integrità dell'entità;
– integrità referenziale.

La regola dell'integrità dell'entità dice che, per ogni istanza di un'entità, il valore della chiave primaria deve esistere, essere unico e non null.

Senza l'integrità nell'entità la chiave primaria non potrebbe essere tale.

La regola dell'integrità referenziale afferma invece che, per ogni valore della chiave esterna, deve esistere un valore di chiave primaria nella tabella associata.

Quindi, se una tabella ha una chiave esterna, allora ogni valore della chiave esterna deve coincidere con il valore della chiave primaria della tabella a cui fa riferimento.

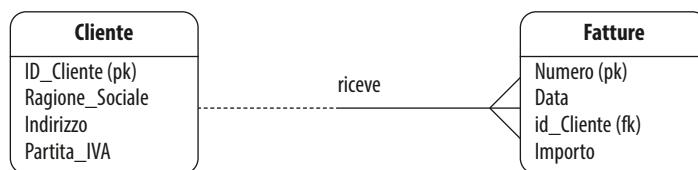
Le operazioni che devono rispettare queste regole sono sostanzialmente l'inserimento e la cancellazione di record, che esamineremo di seguito nel dettaglio.

Regole di inserzione, cancellazione e modifica

Una chiave esterna crea una relazione gerarchica tra due entità associate. L'entità che contiene la chiave esterna si chiama figlio o dipendente e l'entità che contiene la chiave primaria da cui quella esterna è desunta si chiama padre.
Per mantenere l'integrità referenziale tra padre e figlio, mentre i dati vengono inseriti o cancellati dal database, devono essere prese in considerazione alcune regole di inserzione, cancellazione o modifica.
A seconda del DBMS che si sta usando, queste regole possono essere attivate automaticamente oppure per intervento dell'utente.

Nel seguito della trattazione, tali regole sono riportate in ordine di ristrettezza decrescente, a partire da quella "più rigida" fino ad arrivare all'ultima che "permette tutto" e che, quindi, non garantisce il rispetto della integrità referenziale.

Regole di inserzione

Indichiamo di seguito le regole di inserzione comunemente implementate.

1 Inserzione dipendente: consente l'inserzione di un'istanza nell'entità figlio solo se la chiave padre esiste già.

2 Inserzione automatica: permette l'inserzione di un'istanza figlio e, nel caso in cui l'istanza dell'entità padre non esista, viene creata.

3 Inserzione nulla: permette l'inserzione di un'entità figlio e, nel caso che l'istanza padre non esista, la chiave esterna nel figlio viene messa a null.

4 Inserzione di default: permette l'inserzione di un'istanza dell'entità figlio e, se l'istanza dell'entità padre non esiste, la chiave esterna del figlio viene impostata a un valore predefinito.

5 Nessun effetto: questa regola dice che l'inserzione di un'istanza dell'entità figlio è sempre permessa e non è richiesta in alcun modo l'esistenza della istanza padre: quindi non viene fatto alcun controllo di consistenza.

ESEMPIO

Si considerino le entità Cliente e Fattura legate dalla relazione "riceve".

Ogni Cliente può ricevere una o più Fatture; ogni Fattura deve essere inviata a un solo Cliente.

Rappresentiamo ciascuna entità con una tabella e popoliamo le tabelle con alcuni dati.

ID_CLIENTE	RAGIONE SOCIALE	INDIRIZZO	PARTITA_IVA
100	Dadi e viti S.p.A.	Via Piave 15 – Treviso	12345678901
110	Chiodi s.r.l.	Via Po 22 – Ferrara	23456789012
120	Pinze e tenaglie	Piazza Tevere 33 – Firenze	34567890123
130	Tutto per il bricolage	V.le Arno 155 – Roma	45678901234
140	Fai da te S.p.A.	Via Monte Pollino 12 – Potenza	56789012345
150	La casa del cacciavite	Via Aspromonte 8 – Bari	67890123456
160	Chiavi e brugole	Piazza Etna 1 – Sassari	78901234567

riceve

NUMERO	DATA	ID_CLIENTE	IMPORTO
1235	12/10/2017	100	1000,00
1236	12/10/2017	110	1100,00
1237	12/10/2017	130	1200,00
1238	13/10/2017	100	1300,00
1239	13/10/2017	110	1400,00
1240	13/10/2017	140	1500,00

Proviamo ora a inserire due nuove fatture e vediamo che cosa succede rispetto alle regole di inserzione relative all'integrità referenziale.

1241	15/10/2017	140	1010,00
1242	15/10/2017	170	1020,00

Si noti che nella prima fattura la chiave esterna (id_Cliente =140) è presente nella tabella "padre", mentre nella seconda fattura la chiave esterna (id_Cliente = 170) non è presente.

Se si applica la regola 1 dell'inserzione dipendente, il tentativo di inserzione delle due nuove fatture dà luogo al seguente risultato:

NUMERO	DATA	ID_CLIENTE	IMPORTO	
1235	12/10/2017	100	1000,00	
1236	12/10/2017	110	1100,00	
1237	12/10/2017	130	1200,00	
1238	13/10/2017	100	1300,00	
1239	13/10/2017	110	1400,00	
1240	13/10/2017	140	1500,00	
1241	15/10/2017	140	1010,00	Fattura inserita
1242	15/10/2017	170	1020,00	Fattura rifiutata

Se si applica la regola 2 della inserzione automatica, il tentativo di inserzione delle due nuove fatture dà luogo al seguente risultato:

NUMERO	DATA	ID_CLIENTE	IMPORTO	
1235	12/10/2017	100	1000,00	
1236	12/10/2017	110	1100,00	
1237	12/10/2017	130	1200,00	
1238	13/10/2017	100	1300,00	
1239	13/10/2017	110	1400,00	
1240	13/10/2017	140	1500,00	
1241	15/10/2017	140	1010,00	Fattura inserita
1242	15/10/2017	170	1020,00	Fattura inserita

E viene creata una nuova riga nella tabella Clienti.

ID_CLIENTE	RAGIONE SOCIALE	INDIRIZZO	PARTITA_IVA
100	Dadi e viti S.p.A.	Via Piave 15 – Treviso	12345678901
110	Chiodi s.r.l.	Via Po 22 – Ferrara	23456789012
120	Pinze e tenaglie	Piazza Tevere 33 – Firenze	34567890123
130	Tutto per il bricolage	V.le Arno 155 – Roma	45678901234
140	Fai da te S.p.A.	Via Monte Pollino 12 – Potenza	56789012345
150	La casa del cacciavite	Via Aspromonte 8 – Bari	67890123456
160	Chiavi e brugole	Piazza Etna 1 – Sassari	78901234567
170	null	null	null

Se si applica la regola 3 della inserzione nulla, il tentativo di inserzione delle due nuove fatture dà luogo al seguente risultato:

NUMERO	DATA	ID_CLIENTE	IMPORTO	
1235	12/10/2017	100	1000,00	
1236	12/10/2017	110	1100,00	
1237	12/10/2017	130	1200,00	
1238	13/10/2017	100	1300,00	
1239	13/10/2017	110	1400,00	
1240	13/10/2017	140	1500,00	
1241	15/10/2017	140	1010,00	Fattura inserita
1242	15/10/2017	null	1020,00	Fattura inserita

Se si applica la regola 4 della inserzione default, il tentativo di inserzione delle due nuove fatture dà luogo al seguente risultato.

NUMERO	DATA	ID_CLIENTE	IMPORTO	
1235	12/10/2017	100	1000,00	
1236	12/10/2017	110	1100,00	
1237	12/10/2017	130	1200,00	
1238	13/10/2017	100	1300,00	
1239	13/10/2017	110	1400,00	
1240	13/10/2017	140	1500,00	
1241	15/10/2017	140	1010,00	Fattura inserita
1242	15/10/2017	9999	1020,00	Fattura inserita

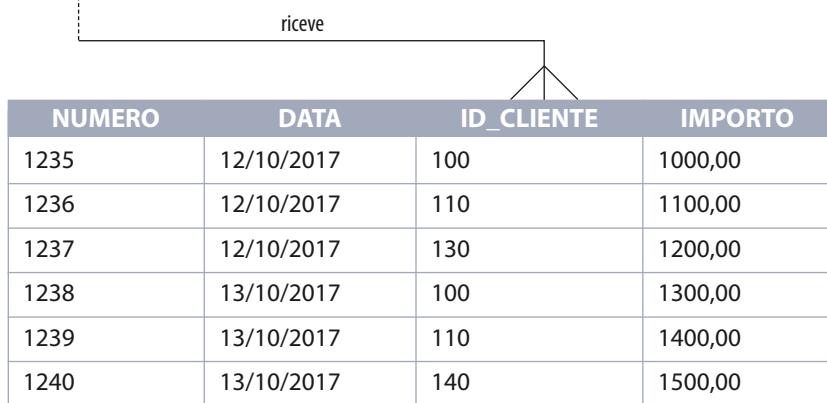
Dove 9999 è un valore inserito per default.

Se si stabilisce che l'inserzione deve essere sempre permessa, il tentativo di inserzione delle due nuove fatture dà luogo al seguente risultato.

NUMERO	DATA	ID_CLIENTE	IMPORTO	
1235	12/10/2017	100	1000,00	
1236	12/10/2017	110	1100,00	
1237	12/10/2017	130	1200,00	
1238	13/10/2017	100	1300,00	
1239	13/10/2017	110	1400,00	
1240	13/10/2017	140	1500,00	
1241	15/10/2017	140	1010,00	Fattura inserita
1242	15/10/2017	170	1020,00	Fattura inserita

Regole di cancellazione

Le regole di cancellazione comunemente implementate sono quelle descritte di seguito.

1. Cancellazione con restrizione: consente la cancellazione dell'istanza dell'entità padre solo se non ci sono istanze dell'entità figlio.
2. Cancellazione a cascata: permette sempre la cancellazione dell'istanza dell'entità padre e cancella tutte le istanze dell'entità figlio corrispondenti.

3 Cancellazione nulla: ammette sempre la cancellazione dell'istanza dell'entità padre e, se esiste un'istanza dell'entità figlio, i valori della chiave esterna per questa istanza sono impostati a null.

4 Cancellazione di default: permette sempre la cancellazione dell'istanza dell'entità padre. Se esiste un'istanza dell'entità figlio, i valori della chiave esterna per questa istanza sono impostati a valori predefiniti.

5 Nessun effetto: questa regola di cancellazione permette sempre la cancellazione e non viene effettuato nessun controllo di consistenza.

ESEMPIO

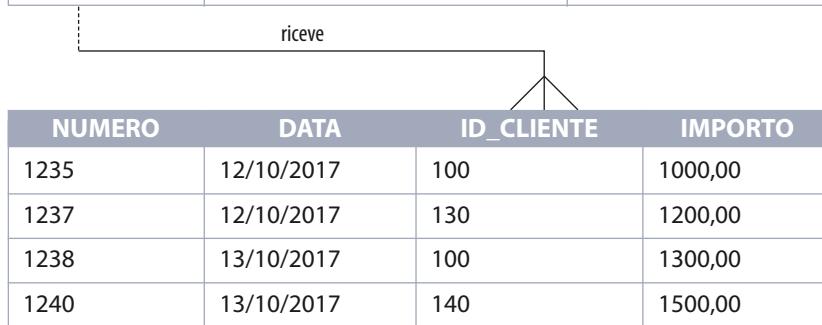
Riprendiamo la struttura dati esaminata in precedenza e vediamo come è possibile eliminare righe dalla tabella Clienti. Non ci sono problemi per la cancellazione di righe dalla tabella delle Fatture in quanto su di essa non vi sono vincoli per l'operazione di cancellazione.

Le tabelle di partenza sono le seguenti:

ID_CLIENTE	RAGIONE SOCIALE	INDIRIZZO	PARTITA_IVA
100	Dadi e viti S.p.A.	Via Piave 15 – Treviso	12345678901
110	Chiodi s.r.l.	Via Po 22 – Ferrara	23456789012
120	Pinze e tenaglie	Piazza Tevere 33 – Firenze	34567890123
130	Tutto per il bricolage	V.le Arno 155 – Roma	45678901234
140	Fai da te S.p.A.	Via Monte Pollino 12 – Potenza	56789012345
150	La casa del cacciavite	Via Aspromonte 8 – Bari	67890123456
160	Chiavi e brugole	Piazza Etna 1 – Sassari	78901234567

riceve

NUMERO	DATA	ID_CLIENTE	IMPORTO
1235	12/10/2017	100	1000,00
1236	12/10/2017	110	1100,00
1237	12/10/2017	130	1200,00
1238	13/10/2017	100	1300,00
1239	13/10/2017	110	1400,00
1240	13/10/2017	140	1500,00

Proviamo ora a eliminare due clienti e vediamo che cosa succede rispetto alle regole di cancellazione relative all'integrità referenziale.
I clienti che vogliamo eliminare sono:

110	Chiodi s.r.l.	Via Po 22 – Ferrara	23456789012
120	Pinze e tenaglie	Piazza Tevere 33 – Firenze	34567890123

Se si applica la regola 1 di cancellazione con restrizione, il cliente con codice 110 non viene eliminato perché nella tabella Fatture sono presenti istanze con chiavi esterne uguali a 110, mentre il cliente con codice 120 può essere eliminato perché nella tabella Fatture non vi è alcun riferimento a esso.

Se si applica la regola 2 della cancellazione a cascata vengono invece eliminati entrambi i clienti, cioè sia quello con codice 110 sia quello con codice 120; inoltre, nella tabella delle Fatture vengono cancellate le istanze con chiave esterna uguale a 110.
Pertanto, la situazione finale dopo l'operazione di cancellazione risulta la seguente:

ID_CLIENTE	RAGIONE SOCIALE	INDIRIZZO	PARTITA_IVA
100	Dadi e viti S.p.A.	Via Piave 15 – Treviso	12345678901
130	Tutto per il bricolage	V.le Arno 155 – Roma	45678901234
140	Fai da te S.p.A.	Via Monte Pollino 12 – Potenza	56789012345
150	La casa del cacciavite	Via Aspromonte 8 – Bari	67890123456
160	Chiavi e brugole	Piazza Etna 1 – Sassari	78901234567

riceve

NUMERO	DATA	ID_CLIENTE	IMPORTO
1235	12/10/2017	100	1000,00
1237	12/10/2017	130	1200,00
1238	13/10/2017	100	1300,00
1240	13/10/2017	140	1500,00

Sono state eliminate le fatture Numero 1236 e 1239.

Se si applica la regola 3 di cancellazione nulla, vengono eliminati entrambi i clienti e nella tabella Fatture le chiavi esterne uguali a 110 vengono sostituite con il valore null.

ID_CLIENTE	RAGIONE SOCIALE	INDIRIZZO	PARTITA_IVA
100	Dadi e viti S.p.A.	Via Piave 15 – Treviso	12345678901
130	Tutto per il bricolage	V.le Arno 155 – Roma	45678901234
140	Fai da te S.p.A.	Via Monte Pollino 12 – Potenza	56789012345
150	La casa del cacciavite	Via Aspromonte 8 – Bari	67890123456
160	Chiavi e brugole	Piazza Etna 1 – Sassari	78901234567

riceve

NUMERO	DATA	ID_CLIENTE	IMPORTO
1235	12/10/2017	100	1000,00
1236	12/10/2017	null	1100,00
1237	12/10/2017	130	1200,00
1238	13/10/2017	100	1300,00
1239	13/10/2017	null	1400,00
1240	13/10/2017	140	1500,00

Se si applica la regola 4 di cancellazione di default, vengono eliminati entrambi i clienti e nella tabella Fatture le chiavi esterne uguali a 110 vengono sostituite con un valore prestabilito.

ID_CLIENTE	RAGIONE SOCIALE	INDIRIZZO	PARTITA_IVA
100	Dadi e viti S.p.A.	Via Piave 15 – Treviso	12345678901
130	Tutto per il bricolage	V.le Arno 155 – Roma	45678901234
140	Fai da te S.p.A.	Via Monte Pollino 12 – Potenza	56789012345
150	La casa del cacciavite	Via Aspromonte 8 – Bari	67890123456
160	Chiavi e brugole	Piazza Etna 1 – Sassari	78901234567

riceve

NUMERO	DATA	ID_CLIENTE	IMPORTO
1235	12/10/2017	100	1000,00
1236	12/10/2017	9999	1100,00
1237	12/10/2017	130	1200,00
1238	13/10/2017	100	1300,00
1239	13/10/2017	9999	1400,00
1240	13/10/2017	140	1500,00

Se si stabilisce che la cancellazione deve essere sempre permessa, il tentativo di eliminazione dei due clienti dà luogo al seguente risultato per la tabella Clienti.

ID_CLIENTE	RAGIONE SOCIALE	INDIRIZZO	PARTITA_IVA
100	Dadi e viti S.p.A.	Via Piave 15 – Treviso	12345678901
130	Tutto per il bricolage	V.le Arno 155 – Roma	45678901234
140	Fai da te S.p.A.	Via Monte Pollino 12 – Potenza	56789012345
150	La casa del cacciavite	Via Aspromonte 8 – Bari	67890123456
160	Chiavi e brugole	Piazza Etna 1 – Sassari	78901234567

Nessuna modifica alla tabella Fatture.

Quando l'operazione di inserimento o l'operazione di cancellazione sono definite senza permessi, viene a mancare completamente la coerenza e l'affidabilità dei dati. Infatti, in entrambi i casi, nell'entità figlio sono presenti chiavi esterne che non hanno riscontro nell'entità padre.

La scelta di quale regola usare viene determinata da alcune linee guida di base per l'inserzione e la cancellazione:
- evitare l'uso di inserzioni e cancellazioni con regola null: generalmente, l'entità padre in una relazione deve esistere e usare la cancellazione null violerebbe questa regola;
- usare la regola di inserzione automatica o dipendente: solo queste regole mantengono i dati coerenti;
- usare la regola di cancellazione a cascata: questa regola assicura che si mantenga la coerenza dei dati per le chiavi esterne.

Regole di modifica

Le regole di modifica sono identiche alle regole di inserzione, in quanto la modifica può essere vista come un caso particolare dell'inserimento: nelle regole prima descritte, basta inserire il termine modifica al posto del termine inserzione.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 La regola dell'integrità dell'entità richiede:
- **a** che il valore delle chiavi sia non null
- **b** che il valore della chiave primaria sia unico
- **c** che il valore della chiave primaria sia non null
- **d** per ogni valore della chiave esterna un valore di chiave primaria
- **e** che il valore della chiave primaria sia unico e non null

2 La regola dell'integrità referenziale richiede:
- **a** che il valore delle chiavi sia non null
- **b** che il valore della chiave primaria sia unico
- **c** che il valore della chiave primaria sia non null
- **d** per ogni valore della chiave esterna un valore di chiave primaria
- **e** che il valore della chiave primaria sia unico e non null

3 Quale tra le seguenti regole di inserzione non è mai implementata?
- **a** Inserimento dipendente
- **b** Inserimento automatico
- **c** Inserimento referenziale
- **d** Inserzione nulla
- **e** Inserzione di default
- **f** Nessun effetto

4 La regola di inserzione nulla consente:
- **a** l'inserzione di un'entità figlio nulla
- **b** sempre l'inserzione di un'entità figlio
- **c** sempre l'inserzione di un'entità figlio solo se esiste l'istanza padre
- **d** sempre l'inserzione di un'entità figlio solo se il padre è null

5 Quale tra le seguenti regole di cancellazione non è mai implementata?
- **a** Cancellazione con restrizione
- **b** Cancellazione a cascata
- **c** Cancellazione nulla
- **d** Cancellazione multipla
- **e** Cancellazione di default
- **f** Nessun effetto

6 La regola di cancellazione di default consente sempre la cancellazione:
- **a** dell'istanza dell'entità padre
- **b** dell'istanza dell'entità figlio
- **c** dell'istanza dell'entità padre solo se non ha figli
- **d** dell'istanza dell'entità padre solo se non ha padri

VERO/FALSO

1 Solo una chiave primaria può avere valore null. V F
2 In una tabella è possibile avere gli stessi valori in due chiavi esterne.
3 In due tabelle differenti è possibile avere gli stessi valori di chiave primaria.
4 Le regole di inserzione servono per mantenere l'integrità referenziale tra il padre e il figlio. V F
5 La cancellazione di default permette sempre la cancellazione dell'istanza senza controlli V F
6 È consigliato l'uso di inserzioni e cancellazioni con regola null. V F
7 La regola di cancellazione a cascata mantiene la coerenza dei dati per le chiavi esterne. V F
8 La regola di inserzione automatica o dipendente mantiene i dati coerenti. V F

VERIFICA... le conoscenze

COMPLETAMENTO

1 Regole di integrità. Scegli la parola corretta tra le due alternative.
Il valore della chiave primaria *deve/può esistere*.
Il valore della chiave primaria *deve/può* essere unico.
Il valore della chiave primaria deve essere *null/non null*.
Per ogni valore della chiave *esterna/primaria* deve esistere un valore di chiave *esterna/primaria* nella tabella associata.

2 Completa le frasi con le parole di seguito indicate.
default – cascata – restrizione – nulla

La regola di inserimento .... consente l'inserzione di un'istanza nell'entità figlio solo se la chiave padre esiste già.

La regola di inserzione .... permette l'inserzione di un'istanza figlio. Se l'istanza dell'entità padre non esiste, viene creata.

La regola di inserzione .... consente l'inserzione di un'entità figlio. Se l'istanza padre non esiste, la chiave esterna nel figlio viene messa a null.

La regola di inserzione .... permette l'inserzione di un'istanza di entità figlio. Se l'istanza dell'entità padre non esiste, la chiave esterna del figlio viene impostata a un valore predefinito.

3 Completa le frasi con le parole di seguito indicate.
default – cascata – restrizione – nulla

La regola di cancellazione con .... permette la cancellazione dell'istanza dell'entità padre solo se non ci sono istanze dell'entità figlio.

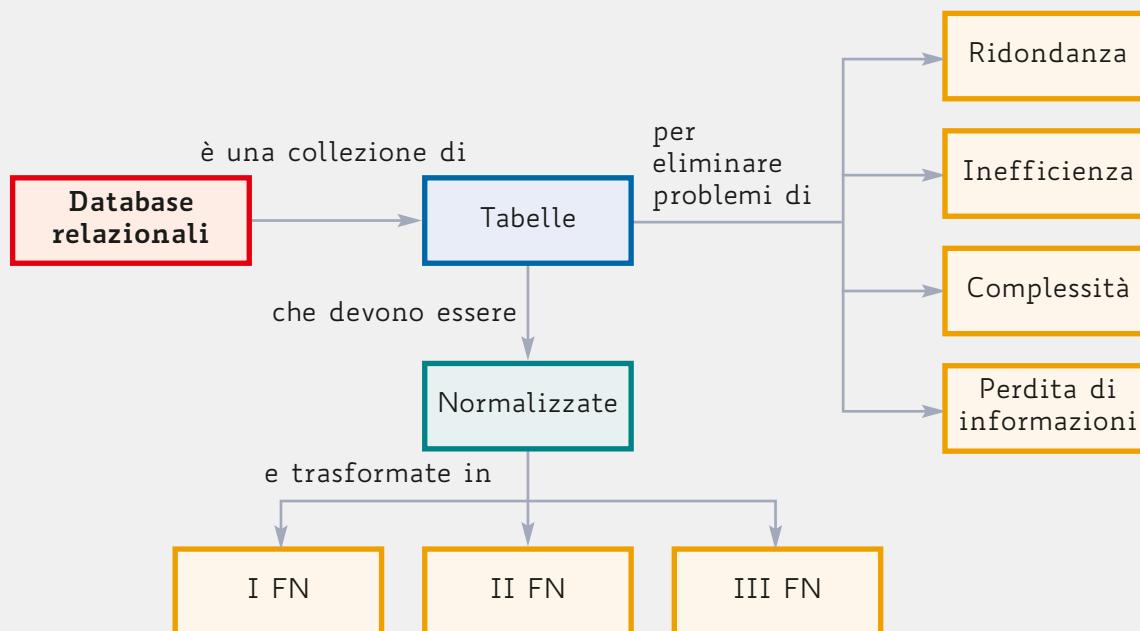
La regola di cancellazione .... consente sempre la cancellazione dell'istanza dell'entità padre e cancella tutte le istanze dell'entità figlio corrispondenti.

La regola .... permette sempre la cancellazione dell'istanza dell'entità padre. Se esiste un'istanza dell'entità figlio, i valori della chiave esterna per questa istanza sono impostati a null.

La regola di cancellazione di .... consente sempre la cancellazione dell'istanza dell'entità padre. Se esiste un'istanza dell'entità figlio, i valori della chiave esterna per questa istanza sono impostati a valori predefiniti.

4 Completa la frase con le parole di seguito indicate.
a cascata– automatica – dipendente

Per mantenere l'integrità referenziale si deve:

– usare la regola di inserzione .... o ....:

– usare la regola di cancellazione ....

11 La normalizzazione delle tabelle

IN QUESTA LEZIONE IMPAREREMO...

- il concetto di dipendenza funzionale
- le motivazioni alla base della normalizzazione
- ad applicare le regole di normalizzazione

MAPPA CONCETTUALE

Normalizzazione

La normalizzazione è un processo che tende a eliminare la ripetizione dei dati (**ridondanza**) e a migliorarne la consistenza.

Ridondanza
Si ha ridondanza dei dati ogni volta che vengono memorizzati inutilmente dei dati ripetuti.

Si tratta una tecnica di progettazione ampiamente utilizzata nel disegno di database relazionali, con l'obiettivo di scomporre le tabelle in tabelle più piccole e senza anomalie, in modo da avere relazioni più strutturate e ridurre al minimo la ridondanza dei dati.

Un database non normalizzato presenta problemi di:
– ridondanza;
– inefficienza;
– complessità;
– perdita di informazioni.

ESEMPIO

Si consideri la seguente tabella che contiene dati relativi al personale di una scuola; alcuni dei dipendenti presi in esame prestano servizio in scuole diverse.

Cognome	Nome	Funzione	Stipendio_orario	Scuola
Rossi	Mario	Preside	50 €	ITIS G. Verdi
Verdi	Filippo	Docente	50 €	ITIS G. Verdi
Verdi	Filippo	Docente	45 €	Liceo P. Rossi
Gialli	Enrico	Docente	50 €	ITIS G. Verdi
Neri	Piero	Docente	40 €	ITIS G. Verdi
Neri	Piero	Docente	40 €	ITC S. Gialli
Bruni	Maria	Bidello	20 €	ITIS G. Verdi
Verdi	Giuseppe	Bidello	20 €	ITIS G. Verdi

La tabella ha problemi di:
– ridondanza: lo stipendio di ciascun dipendente è **ripetuto** in tutte le tuple (record) a esso relative;
– anomalia di modifica: se lo stipendio di un dipendente varia, è necessario andare a modificarne il valore in tutte le tuple nelle quali è presente;
– anomalia di cancellazione: se un dipendente resta senza scuola, va cancellato dalla tabella;
– anomalia di inserimento: un nuovo dipendente non assegnato ad alcuna scuola non può essere inserito.

Le ultime tre anomalie sono spesso indicate semplicemente come anomalie di aggiornamento, intendendo con aggiornamento una qualunque operazione che produca una variazione dei dati dell'archivio.

Queste anomalie sono presenti perché abbiamo utilizzato un'unica tabella per rappresentare informazioni eterogenee, mentre sarebbe stato opportuno suddividere la tabella in più relazioni. Le regole di normalizzazione ci permettono di passare "automaticamente" da una tabella di partenza generica a un insieme di tabelle esenti dalle anomalie sopra elencate.

Lo scopo della normalizzazione è di creare un insieme di tabelle che siano libere da dati ridondanti e che possano essere modificate senza il rischio di perdere la coerenza dei dati contenuti.

La teoria della normalizzazione è basata sul concetto di forma normale: una tabella relazionale è in una particolare forma normale se soddisfa un certo insieme di vincoli.
Attualmente sono state definite cinque forme normali: in questa lezione tratteremo le prime tre, definite da E.F. Codd.

Dipendenze funzionali

Alla base delle prime tre forme normali vi è il concetto di dipendenza funzionale: per comprenderlo, consideriamo l'esempio seguente, dove vengono rappresentate in una tabella alcune informazioni riguardanti un insieme di persone.

ESEMPIO

Partiamo dal seguente schema relazionale dell'entità Persone:

Persone (ID_Anag(pk), Cognome, Nome, Indirizzo, Località, CAP, Telefono)

Popoliamo quindi la tabella con un insieme di tuple.

PERSONE						
ID_Anag	Cognome	Nome	Indirizzo	Località	CAP	Scuola
1	Bianchi	Mario	Via Garibaldi, 10	Padova	35100	332.332222
2	Cerreti	Giovanni	Via Carducci, 12	Milano	20100	322.112222
3	Rossigni	Giacomo	Piazza Caduti, 20	Torino	10100	399.9932222
4	Doni	Margherita	Via Cavour, 33	Padova	35100	389.1244452
5	Tacchi	Alfio	Viale Trento	Treviso	31100	327.1235676
6	Mella	Oscar	Via Cavour, 33	Padova	35100	389.1244452

Osservando i dati memorizzati nella tabella, si può notare che i valori contenuti nelle colonne Nome, Cognome, Indirizzo e Telefono sono associati con uno e un solo valore della colonna ID_Anag.

In una situazione del genere, si dice che le colonne Nome, Cognome, Indirizzo e Telefono sono funzionalmente dipendenti dalla colonna Id_Anag e la colonna Id_Anag è detta determinante per le colonne Nome, Cognome, Indirizzo e Telefono.

La stessa cosa non si può dire per le colonne Località e Cap, in quanto a valori diversi della colonna ID_Anag corrispondono valori ripetuti nelle colonne Località e Cap (per esempio, le chiavi 1, 4, 6 presentano lo stesso valore per la Località e il Cap).
Se si analizzano le dipendenze funzionali tra le colonne si vede che Cognome, Nome, Località, Telefono dipendono funzionalmente dalla chiave primaria ID_Anag, mentre il Cap dipende da Località.

Una colonna Y di una tabella R viene detta funzionalmente dipendente dalla colonna X di R se ogni valore di X in R viene associato con un solo valore di Y. La colonna X è detta determinante per la colonna Y.

Dire che la colonna Y è funzionalmente dipendente da X è come dire che i valori della colonna X identificano i valori della colonna Y. Se la colonna X è una chiave primaria, allora tutte le colonne nella tabella R devono essere funzionalmente dipendenti da X.

Una notazione breve per descrivere una dipendenza funzionale è la seguente:
R.X → R.Y
Tale notazione può essere letta come: "nella tabella relazionale R la colonna X determina funzionalmente i valori della colonna Y".

Con la normalizzazione si rimuovono i dati ridondanti dalle tabelle decomponendole (dividendole) in tabelle più piccole attraverso operazioni di proiezione. Lo scopo è quello di avere solo chiavi primarie a sinistra delle dipendenze funzionali.

Per essere corretta, la decomposizione non deve provocare una perdita di dati, ovvero le nuove tabelle possono essere ricombinate mediante join naturali per ricreare le tabelle originali senza generare alcun dato spurio o ridondante.

Prima forma normale

Una relazione si dice in prima forma normale (1NF) se e solo se tutti i suoi attributi sono valori atomici: ciò implica che né gli attributi né i valori da questi assunti possono essere scomposti ulteriormente.

Una tabella relazionale, per definizione, è nella prima forma normale. Tutti i valori delle colonne sono atomici, ovvero non sono formati da campi composti.

Vediamo alcuni esempi di applicazione della prima forma normale.

1 La tabella seguente non è in prima forma normale.

Codice_Fiscale	Generalità
RSSLBR79Y12T344A	Rossi Alberto

Infatti l'attributo Generalità è un campo composto: per rendere la tabella in forma normale è sufficiente scomporre tale attributo in due campi (Cognome e Nome).

Codice_Fiscale	Cognome	Nome
RSSLBR79Y12T344A	Rossi	Alberto

2 Analoga situazione si verifica nella tabella seguente:

Codice_Fiscale	Cognome	Nome	Residenza
RSSLBR79Y12T344A	Rossi	Alberto	Via Roma, 22; 22100 Como
BNCGNN69K42Z123A	Bianchi	Gianni	Piazza Lodi, 3; 24100 Bergamo

L'attributo Residenza è infatti un campo composto: trasformiamo la tabella in 1NF dividendo il campo Residenza in tre nuovi attributi (Indirizzo, CAP e Città).

Codice_Fiscale	Cognome	Nome	Indirizzo	CAP	Città
RSSLBR79Y12T344A	Rossi	Alberto	Via Roma, 22	22100	Como
BNCGNN69K42Z123A	Bianchi	Gianni	Piazza Lodi, 3	24100	Bergamo

Anche la tabella ottenuta, tuttavia, non è ancora in 1NF: infatti l'attributo Indirizzo è composto a sua volta da tre possibili attributi: Tipo_Via, Nome_Via, Civico: sarebbe quindi da scomporre in questo modo:

Codice_Fiscale	Cognome	Nome	Tipo_Via	Nome_Via	Civico	CAP	Città
RSSLBR79Y12T344A	Rossi	Alberto	Via	Roma	22	22100	Como
BNCGNN69K42Z123A	Bianchi	Gianni	Piazza	Lodi	3	24100	Bergamo

AREA DIGITALE

Il CAP è un dato non atomico

Anche se formalmente corretta, nella pratica la scomposizione dell'indirizzo nelle sue tre componenti viene effettuata raramente, tranne in applicazioni particolari nelle quali è importante isolare i singoli elementi.

Seconda forma normale

Dopo aver verificato e eventualmente trasformato le tabelle in 1FN si procede con le operazioni di normalizzazione per portare le tabelle anche in 2FN.

Partiamo adesso dal seguente schema relazionale per creare una tabella dove inseriremo i dati di alcuni articoli forniti da diversi fornitori:

Forniture (ID_Fornitore(pk), ID_Articolo(pk), Città, id_Stato(fk), Qtà)

dove:
– ID_Fornitore è il numero identificativo del fornitore (la sua chiave primaria);
– ID_Articolo è il codice identificativo dell'articolo;
– Città è il nome della città dove il fornitore risiede;
– Id_Stato è il codice dello Stato a cui appartiene la Città;
– Qtà è il numero di articoli forniti.

Per associare univocamente la quantità fornita con l'articolo (ID_Articolo) e il fornitore (ID_Fornitore), viene usata una chiave primaria composita di ID_Fornitore e ID_Articolo.

La tabella riportata di seguito mostra un ipotetico contenuto della tabella Forniture in 1NF.

FORNITURE				
ID_Fornitore(pk)	id_Stato(fk)	Città	ID_Articolo(pk)	Qtà
S1	20	Roma	P1	300
S1	20	Roma	P4	150
S1	20	Roma	P5	350
S1	20	Roma	P6	500
S2	10	Londra	P1	400
S2	10	Londra	P2	300
S3	30	Parigi	P2	200
S4	40	Budapest	P2	100
S4	40	Budapest	P4	150
S4	40	Budapest	P5	250

La tabella Forniture, tuttavia, anche se è nella forma 1NF, presenta il difetto di contenere dati ridondanti.

Le informazioni sulla città del fornitore e sullo Stato sono infatti ripetute per ogni articolo fornito.

Sappiamo che la ridondanza provoca ciò che viene chiamata anomalia di aggiornamento: analizziamo quali problemi nascono nelle diverse operazioni di aggiornamento in questo specifico esempio.

Inserimento	L'informazione che un nuovo fornitore (S5) sia residente in una particolare città (Atene) non può essere aggiunta finché non fornisce un articolo.
Cancellazione	Se una riga viene cancellata, allora non solo le informazioni sulla quantità e sull'articolo vengono perdute, ma anche quelle sul fornitore.
Modifica	Se il fornitore S1 si trasferisce da Roma a New York, allora quattro righe devono essere modificate con questa nuova informazione.

Si ottiene un miglioramento trasformando la tabella in seconda forma normale.

Una tabella in seconda forma normale (2NF) è una tabella in 1NF in cui tutte le colonne non chiave sono completamente dipendenti dall'intera chiave primaria: non sono cioè ammesse colonne che dipendono funzionalmente solo da una parte della chiave primaria composta.

La tabella Forniture è in 1NF e non in 2NF perché le colonne id_Stato e Città sono funzionalmente dipendenti solo dalla colonna ID_Fornitore della chiave composita (ID_Fornitore + ID_Articolo). Questo può essere illustrato elencando le dipendenze funzionali nella tabella Forniture:

ID_Fornitore → Città, id_Stato
Città → id_Stato
(ID_Fornitore, ID_Articolo) → Qtà

Di seguito viene illustrata la procedura per trasformare una tabella 1NF in una 2NF.

1. Va identificato ogni determinante diverso dalla chiave composita e le colonne che determina.
2. Va creata e nominata una nuova tabella per ogni determinante e le colonne chiave che determina.
3. Le colonne così individuate vanno spostate dalla tabella originale alla nuova tabella. Il determinante diventa la nuova chiave primaria della nuova tabella.
4. Le colonne che sono state spostate dalla tabella originale vanno cancellate eccetto i determinanti che servono come chiave esterna.
5. La tabella originale può essere rinominata per mantenere un significato semantico.

ESEMPIO

Per trasformare la tabella Forniture in 2NF spostiamo le colonne ID_Fornitore, id_Stato e Città in una nuova tabella chiamata Fornitori.

FORNITORI		
ID_Fornitore(pk)	**id_Stato(fk)**	**Città**
S1	20	Roma
S2	10	Londra
S2	10	Londra
S3	30	Parigi
S4	40	Budapest

La colonna ID_Fornitore diventa la chiave primaria della nuova tabella Fornitori, mentre nella tabella Forniture, ID_Fornitore resta un componente della chiave primaria ma ha anche la funzione di chiave esterna per l'associazione con la tabella Fornitori.

FORNITURE		
ID_Articolo(pk)	**ID_Fornitore(pk)**	**Qtà**
P1	S1	300
P4	S1	150
P5	S1	350
P6	S1	500
P1	S2	400
P2	S2	300
P2	S3	200
P2	S4	100
P4	S4	150
P5	S4	250

Vediamo un altro esempio di applicazione della seconda forma normale.

ESEMPIO

Abbiamo detto che, affinché una base di dati possa essere in 2NF, è necessario che:
– si trovi già in 1NF;
– tutti i campi non chiave dipendano dall'intera chiave primaria (e non da una parte di essa).

Quindi, il presupposto per verificare se il database è in 2NF è che sia presente una chiave primaria composta.
Supponiamo, per esempio, di avere a che fare con il database di una scuola con una chiave primaria composta dai campi Matricola e Codice_Materia.

VALUTAZIONI			
Matricola	**Codice_Materia**	**Alunno**	**Voto_Materia**
1234	Info1	Rossi	6
1234	Italiano1	Rossi	7
2233	10 Italiano1	Verdi	8

Il database sopra riportato si trova in 1NF ma non in 2NF, in quanto il campo Alunno non dipende dall'intera chiave ma solo da una parte di essa (Codice_Matricola).
Per trasformare il nostro database in 2NF dovremo pertanto scomporlo in due tabelle:

VALUTAZIONI		
Matricola	**Codice_Materia**	**Voto_Materia**
1234	Info1	6
1234	Italiano1	7
2233	10 Italiano1	8

ALUNNI	
Matricola	**Alunno**
1234	Rossi
2233	Verdi

Nella prima tabella il campo Voto_Materia dipende correttamente dalla chiave primaria composta da Matricola e Codice_Materia; nella seconda tabella il campo Alunno dipende correttamente dalla sola chiave primaria presente (Matricola).
Ora il nostro database è normalizzato in seconda forma normale.

Vediamo un ultimo esempio.

ESEMPIO

Abbiamo una tabella che rappresenta delle scrivanie da ufficio prodotte da un'azienda, dove la chiave primaria è data da Modello_Tavolo e Tipo_Legno.

SCRIVANIE		
Modello_Tavolo	**Tipo_Legno**	**Tipo_Finitura**
Marina	Mogano	Lisciatura
Marina	Ciliegio	Levigatura
President	Noce	Lucidatura
President	Mogano	Lisciatura
Top	Noce	Lucidatura

Tipo_Finitura è dipendente funzionalmente da un sottoinsieme della chiave primaria, ovvero da Tipo_Legno, dunque la tabella non è in 2NF.
È possibile normalizzare la relazione trasformandola in due tabelle:

SCRIVANIE	
Modello_Tavolo	**Tipo_Legno**
Marina	Mogano
Marina	Ciliegio
President	Noce
President	Mogano
Top	Noce

LAVORAZIONI	
Tipo_Legno	**Tipo_Finitura**
Mogano	Lisciatura
Ciliegio	Levigatura
Noce	Lucidatura

In una tabella dove la chiave primaria è semplice non ci sono problemi inerenti alla seconda forma normale.

Terza forma normale

Riprendiamo ora il primo esempio del paragrafo precedente e esaminiamo le tabelle di Forniture e Fornitori.

Forniture (ID_Articolo(pk), ID_Fornitore(pk), Qtà)
Fornitori (ID_Fornitore(pk), id_Stato(fk), Città)

Le tabelle in 2NF contengono ancora delle anomalie di aggiornamento.

Nella tabella Fornitori tali anomalie sono le seguenti:
– insert: l'inserimento di una nuova città non può essere effettuato finché non c'è un fornitore in quella città;
– delete: cancellare qualsiasi riga dalla tabella Fornitori cancella l'informazione sulla città oltre all'associazione tra città e fornitore.

È pertanto necessario procedere con un nuovo passo di normalizzazione.

La terza forma normale 3FN richiede che la tabella sia già in seconda forma normale e che tutte le colonne in una tabella relazionale siano dipendenti solamente dalla chiave primaria.

Possiamo osservare che la tabella Forniture, dopo la trasformazione in 2NF, è di fatto già in 3NF, dato che la colonna non chiave Qtà è completamente dipendente dalla chiave primaria (ID_Articolo + ID_Fornitore).

La tabella Fornitori è invece ancora in 2NF, perché la colonna id_Stato non dipende dalla chiave ID_Fornitore, ma dalla colonna Città.

In pratica, in questa tabella è presente una dipendenza transitiva.

Una dipendenza transitiva si ha quando una colonna è un determinante di un'altra colonna, che a sua volta è un determinante per altre colonne.
Esprimendola in forma matematica:
se R.X → R.Y → R.Z, R.Z è in dipendenza transitiva rispetto a R.X.

ESEMPIO

Nella tabella Fornitori, id_Stato è in dipendenza transitiva rispetto a ID_Fornitore.

Fornitori.ID_Fornitore → Fornitori.Città
Fornitori.Città → Fornitori.id_Stato

La procedura di trasformazione della tabella in 3NF è la seguente.

1. Va identificato qualsiasi determinante, oltre alla chiave primaria, e le colonne che determina.
2. Va creata e nominata una nuova tabella per ogni determinante e le colonne univoche che determina.
3. Le colonne del determinante vanno spostate dalla tabella originale nella nuova tabella. Il determinante diventa la chiave primaria della nuova tabella.
4. Le colonne che sono state spostate dalla tabella originale vanno cancellate eccetto il determinante che serve come chiave esterna.

ESEMPIO

Per trasformare Fornitori in 3NF creiamo una nuova tabella chiamata Città_Stato e spostiamo le colonne Città e id_Stato in essa: id_Stato viene cancellata dalla tabella originale Fornitori, Città viene lasciata e serve come chiave esterna per mantenere l'associazione tra le tabelle Fornitori e Città_Stato.

FORNITORI	
ID_Fornitore(pk)	Città(fk)
S1	Roma
S2	Londra
S3	Parigi
S4	Budapest

CITTÀ_STATO	
Città(pk)	id_Stato(fk)
Roma	20
Londra	10
Parigi	30
Budapest	40

Il vantaggio di avere una tabella in 3NF consiste nell'eliminazione della ripetizione dei dati in modo da ridurre l'ingombro sulla memoria di massa per il salvataggio delle tabelle e nell'eliminazione delle anomalie da aggiornamento.

ESEMPIO

Supponiamo di avere una base dati di una palestra che associ il codice fiscale dell'iscritto al corso frequentato e all'istruttore di riferimento. Si supponga che il nostro DB abbia un'unica chiave primaria (Codice_Fiscale) e sia così strutturato:

ISCRIZIONI		
Codice_Fiscale	Corso	Istruttore
LBRRSS93Y12T344A	Body building	Marco
GNNBNCT96A11L611B	Body building	Marco
LBRMNN96E64A112A	Body building	Marco
GLSTMT99U66P109B	Fitness	Giovanna

Il database non è certamente 3NF in quanto il campo Istruttore non dipende dalla chiave primaria ma dal campo Corso (che non è chiave).

Per normalizzare il nostro DB in 3NF dovremo scomporlo nelle seguenti due tabelle.

ISCRIZIONI	
Codice_Fiscale	**Corso**
LBRRSS93Y12T344A	Body building
GNNBNCT96A11L611B	Body building
LBRMNN96E64A112A	Body building
GLSTMT99U66P109B	Fitness

ISTRUTTORE	
Corso	**Istruttore**
Body building	Marco
Fitness	Giovanna

Il nostro database è ora in terza forma normale.

Esempio riepilogativo

SITUAZIONE

Si vuole definire la struttura dati di un archivio che deve memorizzare le fatture commerciali.

SOLUZIONE

Come metodologia risolutiva non utilizzeremo i diagrammi E-R ma proveremo ad affrontare il problema con un nuovo approccio, cioè partendo dall'elenco completo di tutti i dati da memorizzare e applicando successivamente in sequenza le regole di normalizzazione alle tabelle che via via vengono individuate.

Le informazioni sui dati da archiviare sono ricavate direttamente da un esempio cartaceo osservando una "semplice" fattura commerciale, dove trascuriamo i dati calcolati in quanto derivati da altri campi e quindi superflui.
Ipotizziamo di realizzare un'unica tabella con i seguenti campi:

Nr_Fattura, Data, Codice_Cli, Ragione_Sociale, Indirizzo, ID_Agente, Nome_Agente, Nr_ Item, Descrizione, Quantità, Prezzo_Unitario (questi ultimi quattro ripetuti più volte)

Prima forma normale

Separiamo ora dalla tabella principale i gruppi di campi che si ripetono, altrimenti avremo delle celle della tabella con più valori, e quindi non in 1NF.
Creiamo una nuova tabella con questi gruppi di campi e la chiamiamo RigheFatt, dato che ogni record costituisce una riga del documento fattura. Risulta essere così strutturata:

RigheFatt (Nr_Fattura(pk), Data(pk), Nr_Item(pk), Descrizione, Quantità, Prezzo_Unitario)

La chiave primaria della nuova tabella è sempre una chiave composta: di solito tra il riferimento al documento (Nr_Fattura e Data) e un campo che identifica univocamente la linea che si ripete, come Nr_Item.
I campi che si ripetono sono rimossi dalla tabella principale che si riduce a:

Fatture (Nr_Fattura(pk), Data(pk), Codice_Cli, Ragione_Sociale, Indirizzo, ID_Agente, Nome_Agente)

Dato che nessun campo è duplicato e inoltre ogni campo è semplice, il nostro archivio è in 1FN.

Seconda forma normale

Occorre adesso rimuovere le dipendenze parziali: creiamo delle tabelle separate con i dati funzionalmente dipendenti e i loro determinanti. Nella tabella:

RigheFatt (Nr_Item(pk), Nr_Fattura(pk), Data(pk), Descrizione, Quantità, Prezzo_Unitario)

c'è la dipendenza parziale tra Nr_Item e Descrizione dato che la descrizione del prodotto dipende unicamente dall'articolo (Nr_Item) e non dal numero della fattura.
Quindi il campo Descrizione viene rimosso dalla tabella originale e la sua chiave viene lasciata nella tabella di partenza come chiave esterna per collegare la tabella RigheFatt con la nuova tabella. Pertanto la tabella RigheFatt si spezza in:

Item (Nr_Item(pk), Descrizione)
RigheFatt (Nr_Item(pk), Nr_Fattura(pk), Data(pk), Quantità, Prezzo_Unitario)

Terza forma normale

Rimuoviamo adesso le dipendenze transitive e creiamo una tabella separata per questa dipendenza funzionale, lasciando una copia del determinante nella tabella originale come chiave esterna. Analizziamo la tabella Fatture:

Fatture (Nr_Fattura(pk), Data(pk), Codice_Cli, Ragione_Sociale, Indirizzo, ID_Agente, Nome_Agente)

Abbiamo due dipendenze funzionali:
– Ragione_Sociale e Indirizzo dipendono solo da Codice_Cli;
– Nome_Agente dipende unicamente da ID_Agente.

I loro determinanti Codice_Cli e ID_Agente non fanno parte della chiave primaria, né sono chiave: quindi creiamo le due nuove tabelle.

Clienti (Codice_Cli(pk), Ragione_Sociale, Indirizzo)
Agenti (ID_Agente(pk), Nome_Agente)

Tutti questi campi, a eccezione delle loro chiavi primarie, sono rimossi dalla tabella Fatture che diventa:

Fatture (Nr_Fattura(pk), Data(pk), Codice_Cli, id_Agente(fk))

Il nostro archivio normalizzato è costituito dalle cinque tabelle seguenti:

Clienti (Codice_Cli(pk), Ragione_Sociale, Indirizzo)
Agenti (ID_Agente(pk), Nome_Agente)
Fatture (Nr_Fattura(pk), Data(pk), Codice_Cli, id_Agente(fk))
RigheFatt (Nr_Item(pk), Nr_Fattura(pk), Data(pk), Quantità, Prezzo_Unitario)
Item (Nr_Item(pk), Descrizione)

Allo stesso risultato si può giungere partendo dal modello concettuale dei dati E-R e applicando le regole di semplificazione delle relazioni. Per esempio, si può partite individuando le seguenti entità: Fatture, Item, Agenti, Clienti. La relazione tra Fatture e Item è del tipo n, n e viene semplificata introducendo la relazione ponte RigheFatt.

Una volta descritte le entità secondo lo schema relazionale si verifica se tutte le tabelle sono normalizzate e si ottiene il medesimo risultato.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Un database normalizzato non presenta problemi di:
- a ridondanza
- b inefficienza
- c correttezza
- d complessità
- e perdita di informazioni

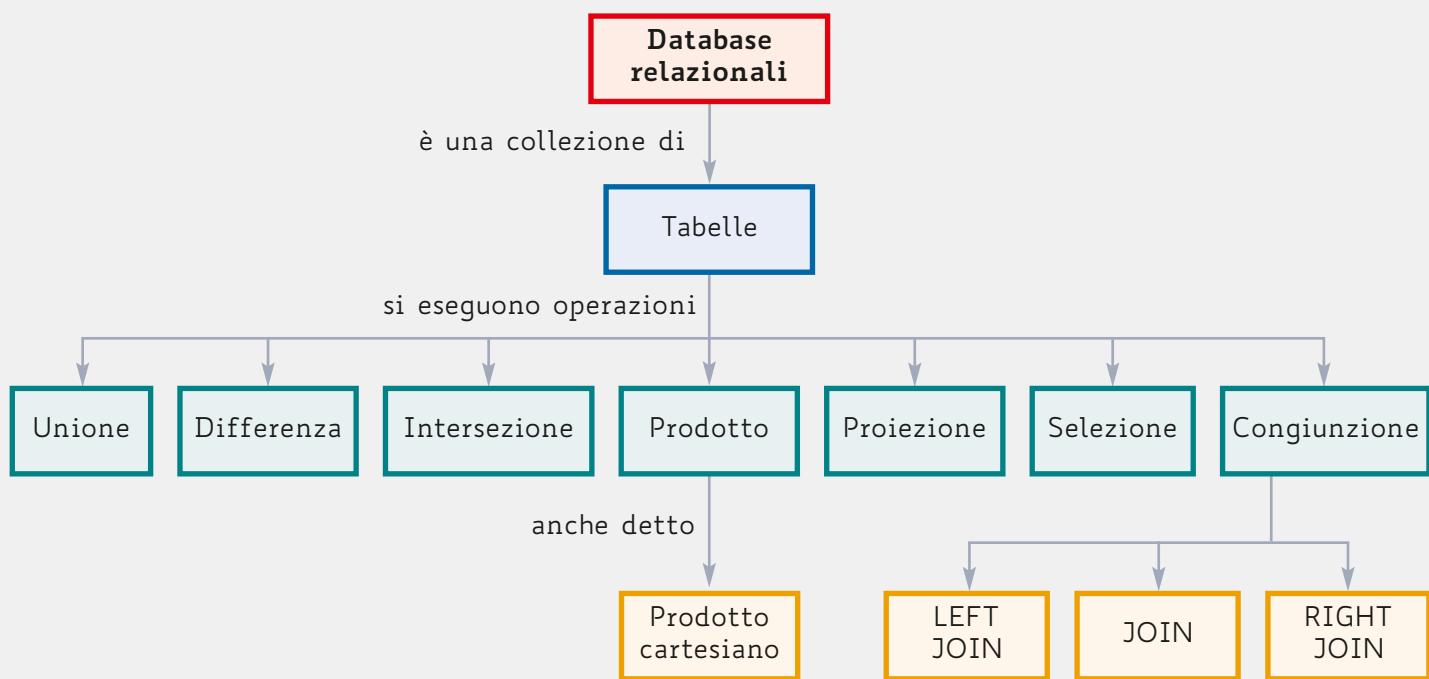
2 In una tabella possiamo riscontrare:
- a anomalia di modifica
- b anomalia di ricerca
- c anomalia di cancellazione
- d anomalia di inserimento

3 Con aggiornamento si intende:
- a la modifica di un'informazione
- b la modifica in un archivio
- c la sostituzione di un dato
- d la modifica di una tabella

4 Una relazione per essere in seconda forma normale (2NF) deve: *(3 risposte)*
- a essere in 1NF
- b avere la chiave primaria semplice
- c avere la chiave esterna e funzionalmente dipendente
- d avere le colonne che dipendono dalla intera chiave primaria

5 Una relazione per essere in terza forma normale (3NF) deve: *(3 risposte)*
- a essere in 1NF
- b essere in 2NF
- c avere tutte le colonne dipendenti solamente
- d dalla chiave primaria
- e avere tutte le colonne dipendenti solamente dalla chiave esterna

VERIFICA... le competenze

PROBLEMI

Normalizza le seguenti relazioni complesse, indicando per ogni passo i benefici ottenuti e le anomalie ancora presenti.

1 Anagrafica (nome, cognome, indirizzo, CAP, citta, provincia, regione, telefono, nazione)
2 Interrogazioni (materia, voto, data, nome, cognome, classe, sezione)
3 Professori (cognome, nome, indirizzo, citta, CAP, classe, sezione, materia)
4 CD Musicali (gruppo, titolo, canzone1, canzone2, … canzoneN, genere, lingua)
5 Fatture (cliente, indirizzo, nr_Item, numero_Fat, data, descrizione, quantità, prezzo_Unitario)
6 Animali (cod_Anagrafico, razza, nome_Proprio, sesso, data_Nascita, luogo_Nascita, nazione)
7 Dipendenti (matricola, nominativo, stipendio, progetto, cod_Bilancio, funzione, sede)
8 Fornitori (fornitore, indirizzo, numero_Fat, data, descrizione, quantità, prezzo_Unitario)
9 Alunni (cognome, nome, indirizzo, citta, CAP, classe, sezione, specializzazione)
10 Gite (classe, sezione, specializzazione, periodo, destinazione, accompagnatore)
11 Automobili (targa, modello, casa_Automob, anno_Immatric, colore, cilindrata, categoria)
12 Francobolli (nazione, tematica, num_Catalogo, valore_Nuovo, valore_Usato, serie, periodo)
13 Contravvenzioni (cod_Multa, cod_Agente, costo_Multa, targa, cod_Modello, desc_Modello, cod_Marca, desc_Marca, nome_Agente, data, ora, desc_Multa)
14 Giocatori (nazione, squadra, ruolo, cognome, nome, citta_Squadra, citta_Nascita, numero)

12 Operazioni relazionali

IN QUESTA LEZIONE IMPAREREMO...

- a utilizzare gli operatori relazionali
- a costruire nuove tabelle a partire dalle tabelle iniziali

MAPPA CONCETTUALE

didattica inclusiva

Database relazionali
è una collezione di
Tabelle
si eseguono operazioni
Unione
Differenza
Intersezione
Prodotto
Proiezione
Selezione
Congiunzione
anche detto
Prodotto cartesiano
LEFT JOIN
JOIN
RIGHT JOIN

Manipolazione di dati relazionali

Le tabelle relazionali sono insiemi. Le righe delle tabelle possono essere considerate come elementi dell'insieme.
Le operazioni che possono essere eseguite sugli insiemi possono pertanto essere applicate anche alle tabelle relazionali.
Nei paragrafi che seguono descriveremo le otto operazioni eseguibili sulle tabelle relazionali.

Unione

L'operazione di unione di due tabelle relazionali si effettua aggiungendo in coda (*append*) le righe della prima tabella a quelle della seconda tabella per produrre una terza tabella.
Le righe duplicate vengono eliminate.

La notazione per l'unione delle tabelle A e B è A UNION B. Le tabelle usate nell'operazione di unione devono essere compatibili con questo tipo di operazione. Le tabelle compatibili devono avere lo stesso numero di colonne e le colonne corrispondenti devono avere lo stesso dominio.

ESEMPIO

Notate che le righe duplicate [1, A, 2] sono state rimosse.

TABELLA A		
K	X	Y
1	A	2
2	B	4
3	C	6

TABELLA B		
K	X	Y
1	A	2
4	D	8
5	E	10

TABELLA (A UNION B)		
K	X	Y
1	A	2
2	B	4
3	C	6
4	D	8
5	E	10

Se le relazioni hanno nomi di attributi diversi, nella relazione risultato per convenzione si usano i nomi della prima relazione.

ESEMPIO

Nell'esempio seguente la tabella risultato mantiene nella prima colonna il nome del campo Matricola della tabella AM.

ALUNNI MAGGIORENNI (AM)		
Matricola	Cognome	Nome
1	Rossi	Mario
2	Verdi	Luca
3	Bianchi	Luigi

ALUNNI RIPETENTI (AR)		
Matricola	Cognome	Nome
11	Gialli	Mario
2	Verdi	Luca
31	Manzoni	Pino

(AM) UNION (AR)		
Matricola	Cognome	Nome
1	Rossi	Mario
2	Verdi	Luca
3	Bianchi	Luigi
11	Gialli	Mario
31	Manzoni	Pino

Differenza

La differenza di due tabelle è una terza tabella che contiene le righe che esistono nella prima tabella ma che non compaiono nella seconda.

L'operazione di differenza richiede che le tabelle siano compatibili, così come già richiesto per l'unione. Come per l'aritmetica, l'ordine delle sottrazioni ha significato, ovvero, A – B è diverso da B – A.

ESEMPIO

TABELLA A		
K	X	Y
1	A	2
2	B	4
3	C	6

TABELLA B		
K	X	Y
1	A	2
4	D	8
5	E	10

TABELLA A – B		
K	X	Y
2	B	4
3	C	6

TABELLA A		
K	X	Y
1	A	2
2	B	4
3	C	6

TABELLA B		
K	X	Y
1	A	2
4	D	8
5	E	10

TABELLA B – A		
K	X	Y
4	D	8
5	E	10

Vediamo un secondo esempio.

ESEMPIO

ALUNNI RIPETENTI (AR)		
Matricola	Cognome	Nome
1	Rossi	Mario
2	Verdi	Luca
3	Bianchi	Luigi

ALUNNI MOTORIZZATI (AM)		
Matricola	Cognome	Nome
11	Gialli	Mario
2	Verdi	Luca
31	Manzoni	Pino

La differenza delle due tabelle AR – AM genera la seguente tabella, che contiene gli "alunni ripetenti che non sono motorizzati". Alla tabella AR vengono infatti tolti gli alunni motorizzati.

AR – AM		
Matricola	Cognome	Nome
1	Rossi	Mario
3	Bianchi	Luigi

Scambiando l'ordine, la differenza tra le due tabelle AM – AR genera la seguente tabella che contiene gli "alunni motorizzati che non sono ripetenti". Infatti alla tabella alunni motorizzati vengono tolti gli alunni ripetenti.

AM – AR		
Matricola	Cognome	Nome
11	Gialli	Mario
31	Manzoni	Pino

Intersezione

L'intersezione di due tabelle è una terza tabella che contiene le righe comuni a entrambe. Le tabelle devono essere compatibili come per l'unione e la differenza.

La notazione per l'intersezione di A e B è A INTERSEC B.

ESEMPIO

TABELLA A		
K	X	Y
1	A	2
2	B	4
3	C	6

TABELLA B		
K	X	Y
1	A	2
4	D	8
5	E	10

A INTERSEC B		
K	X	Y
1	A	2

Riprendiamo adesso il precedente esempio degli alunni per individuare quelli ripetenti e motorizzati.

ESEMPIO

ALUNNI RIPETENTI (AR)		
Matricola	Cognome	Nome
23	Rossi	Mario
32	Verdi	Luca
43	Bianchi	Luigi

ALUNNI MOTORIZZATI (AM)		
Matricola	Cognome	Nome
54	Gialli	Mario
32	Verdi	Luca
12	Manzoni	Pino

L'intersezione delle due tabelle genera la seguente tabella che contiene gli "alunni ripetenti motorizzati".

AM ∩ AR		
Matricola	Cognome	Nome
32	Verdi	Luca

Prodotto

Il prodotto di due tabelle, chiamato anche prodotto cartesiano, è la concatenazione di ogni riga della prima tabella con ogni riga della seconda tabella.

Il prodotto della tabella A (che ha m righe) e la tabella B (che ha n righe) è la tabella C (che ha m * n righe). Il prodotto è denotato come A × B.

ESEMPIO

TABELLA A		
K	X	Y
1	A	2
2	B	4
3	C	6

TABELLA B		
K	X	Y
1	A	2
4	D	8
5	E	10

A x B					
A.k	A.x	A.y	B.k	B.x	B.y
1	A	2	1	A	2
1	A	2	4	D	8
1	A	2	5	E	10
2	B	4	1	A	2
2	B	4	4	D	8
2	B	4	5	E	10
3	C	6	1	A	2
3	C	6	4	D	8
3	C	6	5	E	10

Le intestazioni delle colonne A.k … B.y devono essere lette nel modo seguente:

A.k = colonna k della tabella A, … B.y = colonna y della tabella B

L'operazione di prodotto non è per se stessa molto utile, comunque è spesso usata come passo intermedio di una JOIN (*vedi in seguito*).

ESEMPIO

ANIMALI	
Codice A	**Nome A**
1	Cane
2	Gatto
3	Pecora

COLORI
Descrizione C
Giallo
Rosso
Verde

Il prodotto delle due tabelle precedenti genera la seguente tabella:

ANIMALI X COLORI		
Codice A	**Nome A**	**Descrizione C**
1	Cane	Giallo
1	Cane	Rosso
1	Cane	Verde
2	Gatto	Giallo
2	Gatto	Rosso
2	Gatto	Verde
3	Pecora	Giallo
3	Pecora	Rosso
3	Pecora	Verde

Le operazioni di manipolazione delle tabelle viste finora hanno scarsa utilità nei problemi applicativi e servono principalmente per mostrare le affinità tra l'algebra degli insiemi e l'algebra relazionale. Le prossime tre operazioni sulle tabelle, invece, sono quelle che più frequentemente sono utilizzate nelle applicazioni reali.

Proiezione

L'operatore di proiezione recupera un sottoinsieme di colonne da una tabella.

ESEMPIO

TABELLA A		
K	**X**	**Y**
1	A	2
2	A	4
3	B	6
4	A	8
5	C	10
6	C	12

Proiezione delle colonne K e Y

RISULTATO	
K	**Y**
1	2
2	4
3	6
4	8
5	10
6	12

Selezione

L'operatore di selezione, a volte chiamato selezione ristretta per evitare la confusione con il comando SQL SELECT, recupera un sottoinsieme di righe da una tabella basandosi su una condizione imposta ai valori contenuti in una o più colonne.

ESEMPIO

TABELLA A		
K	X	Y
1	A	2
2	A	4
3	B	6
4	A	8
5	C	10
6	C	12

Selezione condizione: X = A

RISULTATO		
K	X	Y
1	A	2
2	A	4
4	A	8

Si noti che la proiezione riduce il numero di colonne (il grado), ma lascia inalterato il numero di righe (la cardinalità), mentre la selezione riduce il numero di righe, ma lascia inalterato il numero di colonne.

Congiunzione (JOIN)

L'operazione di JOIN combina le operazioni di prodotto e di selezione. L'operatore di JOIN esegue inizialmente un'operazione di prodotto tra due tabelle, successivamente seleziona dalla tabella risultante le righe che soddisfano una determinata condizione tra le colonne della prima e della seconda tabella.

Solitamente la condizione è un'operazione di uguaglianza. I criteri di selezione implicano una relazione tra le colonne della tabella relazionale. Se un criterio di JOIN è basato sull'equivalenza dei valori di una colonna, il risultato viene detto EQUI JOIN.

Una JOIN naturale consiste in una EQUI JOIN in cui le colonne ridondanti vengono eliminate.

L'esempio che segue illustra l'operazione di JOIN. Sulle tabelle D ed E viene eseguita un'operazione di congiunzione confrontando i valori della colonna K che è comune alle due tabelle. In una prima fase viene effettuato un prodotto cartesiano e su questo viene eseguita una selezione per i valori di K della tabella D uguali ai valori di K nella tabella E.

ESEMPIO

TABELLA D		
K	X	Y
1	A	2
2	B	4
3	C	6
4	D	8
5	E	10

TABELLA E	
K	Z
1	20
4	24
5	28
7	32

D X E				
D.k	D.x	D.y	E.k	E.z
1	A	2	1	20
1	A	2	4	24
1	A	2	5	28
1	A	2	7	32
		...		
5	E	10	4	24
5	E	10	5	28
5	E	10	7	32

Selezione per D.k = E.k

EQUI JOIN				
D.k	D.x	D.y	E.k	E.z
1	A	2	1	20
4	D	8	4	24
5	E	10	5	28

JOIN NATURALE			
D.k	D.x	D.y	E.z
1	A	2	20
4	D	8	24
5	E	10	28

Vediamo un altro esempio.

ESEMPIO

GIOCATORI (G)		
Cognome	Nome	CodSquadra
Totti	Francesco	1
Zanetti	Javier	2
Del Piero	Alex	3
Icardi	Mauro	2

SQUADRE (S)	
CodSquadra	Nome
1	Roma
2	Inter
3	Juventus

Eseguiamo dapprima il prodotto delle due tabelle ottenendo:

GIOCATORI X SQUADRE				
G.Cognome	G.Nome	G.CodSquadra	S.CodSquadra	S.Nome
Totti	Francesco	1	1	Roma
Zanetti	Javier	2	1	Roma
Del Piero	Alex	3	1	Roma
Icardi	Mauro	2	1	Roma
Totti	Francesco	1	2	Inter
Zanetti	Javier	2	2	Inter
Del Piero	Alex	3	2	Inter
Icardi	Mauro	2	2	Inter
Totti	Francesco	1	3	Juventus
Zanetti	Javier	2	3	Juventus
Del Piero	Alex	3	3	Juventus
Icardi	Mauro	2	3	Juventus

Effettuiamo ora la selezione per G.CodSquadra = S.Codsquadra e otteniamo la EQUI JOIN.

EQUI JOIN				
G.Cognome	G.Nome	G.CodSquadra	S.CodSquadra	S.Nome
Totti	Francesco	1	1	Roma
Zanetti	Javier	2	2	Inter
Icardi	Mauro	2	2	Inter
Del Piero	Alex	3	3	Juventus

Le due colonne contenenti il codice squadra sono uguali; eseguendo una natural JOIN, una di queste colonne verrebbe automaticamente eliminata, ottenendo:

EQUI JOIN			
G.Cognome	**G.Nome**	**G.CodSquadra**	**S.Nome**
Totti	Francesco	1	Roma
Zanetti	Javier	2	Inter
Icardi	Mauro	2	Inter
Del Piero	Alex	3	Juventus

LEFT JOIN e RIGHT JOIN

L'operazione di JOIN, prevede due varianti:
– LEFT JOIN, che conserva tutte le righe della tabella di sinistra
– RIGHT JOIN, che conserva tutte righe della tabella di destra.

ESEMPIO

GIOCATORI (G)		
Cognome	**Nome**	**CodSquadra**
Totti	Francesco	1
Zanetti	Javier	2
Del Piero	Alex	3
Icardi	Mauro	2
Ronaldo	Cristiano	5

SQUADRE (S)	
CodSquadra	**Nome**
1	Roma
2	Inter
3	Juventus
4	Milan

LEFT JOIN			
G.Cognome	**G.Nome**	**G.CodSquadra**	**S.Nome**
Totti	Francesco	1	Roma
Zanetti	Javier	2	Inter
Del Piero	Alex	3	Juventus
Icardi	Mauro	2	Inter
Ronaldo	Cristiano	5	

RIGHT JOIN			
G.Cognome	**G.Nome**	**G.CodSquadra**	**S.Nome**
Totti	Francesco	1	Roma
Zanetti	Javier	2	Inter
Del Piero	Alex	3	Juventus
Icardi	Mauro	2	Inter
			Milan

Esempi riepilogativi

Riprendiamo in considerazione i dati del database bibliografico della lezione 9 e facciamo alcuni esempi di applicazione degli operatori relazionali per comprendere le potenzialità da essi offerte.

Riproduciamo quindi graficamente lo schema logico popolando le tabelle.

AUTORI					
ID_Autore	Cognome	Nome	Indirizzo	Località	Cap
172	Bianchi	Mario	Via Garibaldi, 10	Brescia	25100
156	Cerreti	Giovanni	Via Carducci, 12	Milano	20100
268	Rossigni	Giacomo	Piazza Caduti, 20	Torino	10100
884	Tacchi	Alfio	Viale Trento	Treviso	31100
343	Bertozzi	Filippo	Via del lago, 9	Arenzano	16011
298	Sommo	Norberto	Largo Roma, 3	Como	22100

AUTORI_TITOLI	
ID_Autore	ID_Titolo
172	7771010
156	7771213
268	7772123
884	7773244
343	7771356
298	7774744

TITOLI				
ID_Titolo	Titolo	Tipo	Prezzo	id_CasaEd
7771010	*C++*	Informatica	20.00	91234
7771213	*La cucina mediterranea*	Cucina	18.50	92233
7772123	*Aumentare gli utili*	Economia	36.70	94354
7773244	*L'ultima neve*	Thriller	15.20	95555
7771356	*Stress: il nostro nemico*	Psicologia	18.40	94354
7774744	*Le leggi di Mendel*	Biologia	21.00	95555

CASE EDITRICI		
ID_CasaEd	Ragione Sociale	Città
91234	Tecniche OK	Brescia
92233	Nuove Gastronomie	Milano
92333	Hoepli	Milano
94354	Mondo Libri	Torino
95555	New Gothics	Treviso

Creare una tabella che contenga titoli e tipi dei libri

A partire dalla tabella Titoli devono essere "estratte" solo due colonne, quindi eseguiamo la proiezione delle colonne Titolo e Tipo.

TITOLI				
ID_Titolo	Titolo	Tipo	Prezzo	id_CasaEd
7771010	*C++*	Informatica	20.00	91234
7771213	*La cucina mediterranea*	Cucina	18.50	92233
7772123	*Aumentare gli utili*	Economia	36.70	94354
7773244	*L'ultima neve*	Thriller	15.20	95555
7771356	*Stress: il nostro nemico*	Psicologia	18.40	94354
7774744	*Le leggi di Mendel*	Biologia	21.00	95555

TITOLI	
Titolo	Tipo
C++	Informatica
La cucina mediterranea	Cucina
Aumentare gli utili	Economia
L'ultima neve	Thriller
Stress: il nostro nemico	Psicologia
Le leggi di Mendel	Biologia

Creare una tabella che contenga le case editrici con sede a Milano

Per ottenere questa tabella è sufficiente eseguire sulla tabella Case_Editrici una selezione per Città = Milano.

CASE EDITRICI		
ID_CasaEd	Ragione Sociale	Città
91234	Tecniche OK	Brescia
92233	Nuove Gastronomie	Milano
92333	Hoepli	Milano
94354	Mondo Libri	Torino
95555	New Gothics	Treviso

E si ottiene:

CASE EDITRICI CITTÀ = MILANO		
ID_CasaEd	Ragione Sociale	Città
92233	Nuove Gastronomie	Milano
92333	Hoepli	Milano

Creare una tabella che contenga per ogni libro la casa editrice

Si deve realizzare una congiunzione tra la tabella Titoli e la tabella Case_Editrici imponendo la condizione Titoli.id_CasaEd = ID_CasaEd.

TITOLI				
ID_Titolo	Titolo	Tipo	Prezzo	id_CasaEd
7771010	*C++*	Informatica	20.00	91234
7771213	*La cucina mediterranea*	Cucina	18.50	92233
7772123	*Aumentare gli utili*	Economia	36.70	94354
7773244	*L'ultima neve*	Thriller	15.20	95555
7771356	*Stress: il nostro nemico*	Psicologia	18.40	94354
7774744	*Le leggi di Mendel*	Biologia	21.00	95555

CASE EDITRICI		
ID_CasaEd	Ragione Sociale	Città
91234	Tecniche OK	Brescia
92233	Nuove Gastronomie	Milano
92333	Hoepli	Milano
94354	Mondo Libri	Torino
95555	New Gothics	Treviso

Si ottiene come risultato la seguente tabella:

RISULTATO						
ID_Titolo	Titolo	Tipo	Prezzo	id_CasaEd	Ragione Sociale	Città
7771010	*C++*	Informatica	20.00	91234	Tecniche OK	Brescia
7771213	*La cucina mediterranea*	Cucina	18.50	92233	Nuove Gastronomie	Milano
7772123	*Aumentare gli utili*	Economia	36.70	94354	Mondo Libri	Torino
7773244	*L'ultima neve*	Thriller	15.20	95555	New Gothics	Treviso
7771356	*Stress: il nostro nemico*	Psicologia	18.40	94354	Mondo Libri	Torino
7774744	*Le leggi di Mendel*	Biologia	21.00	95555	New Gothics	Treviso

"Quale casa editrice ha pubblicato il libro di Sommo Norberto?"

Per rispondere al quesito si deve eseguire una sequenza di operazioni relazionali raggruppate in tre passi in quanto, per riuscire a individuare la casa editrice, di deve "passare" attraverso il libro:
- primo passo: collegare l'autore al titolo di un libro individuando il suo identificatore id_Titolo;
- secondo passo: collegare il libro alla casa editrice individuando il suo identificatore id_CasaEd;
- terzo passo: del codice della casa editrice ora è possibile recuperare i dati completi (Ragione sociale, Città).

Il primo passo si compone di tre operazioni:
- una congiunzione delle tabelle Autori e Autori_Titoli per Autori.ID_Autore = Autori_Titoli.id_Autore;
- una proiezione su Cognome, Nome, ID_Autore, id_Titolo;
- una selezione per Cognome = "Sommo".

Si ottiene in questo modo una nuova tabella che viene chiamata Passo1.

PASSO 1			
Cognome	**Nome**	**ID_Autore**	**id_Titolo**
Sommo	Norberto	298	7774744

Il secondo passo esegue:
- una congiunzione tra la tabella Passo1 e la tabella Titoli per Passo1.id_Titolo = ID_Titolo;
- una proiezione su Cognome, Nome, id_Autore, Id_Titolo, Id_CasaEd.

Si ottiene, così, la tabella Passo2.

PASSO 2				
Cognome	**Nome**	**ID_Autore**	**id_Titolo**	**id_CasaEd**
Sommo	Norberto	298	7774744	95555

Il terzo passo esegue una congiunzione tra la tabella Passo2 e la tabella Case Editrici, che riportiamo di seguito.

CASE EDITRICI		
ID_CasaEd	**Ragione Sociale**	**Città**
91234	Tecniche OK	Brescia
92233	Nuove Gastronomie	Milano
92333	Hoepli	Milano
94354	Mondo Libri	Torino
95555	New Gothics	Treviso

per Passo2.id_CasaEd = CaseEditrici.ID_CasaEd, si ottiene, così, la tabella Risultato, che contiene l'informazione richiesta.

RISULTATO						
Cognome	**Nome**	**ID_Autore**	**id_Titolo**	**id_CasaEd**	**Ragione Sociale**	**Città**
Sommo	Norberto	298	7774744	95555	New Gothics	Treviso

VERIFICA... le conoscenze

SCELTA MULTIPLA

Per ciascun quesito indica qual è l'affermazione falsa.

1 L'operazione di unione di due tabelle relazionali:
a è formata aggiungendo in coda le righe di una tabella a quelle di una seconda
b produce una terza tabella
c copia le righe duplicate
d è permessa se le tabelle hanno lo stesso numero di colonne
e si può eseguire se le colonne corrispondenti hanno lo stesso dominio

2 La differenza di due tabelle relazionali:
a è una terza tabella che contiene le righe nella prima ma non nella seconda
b richiede che le tabelle abbiano lo stesso numero di righe
c richiede che le tabelle siano compatibili per l'unione
d dipende dall'ordine nella sottrazione ha significato cioè A-B è diverso da B-A

3 L'intersezione di due tabelle relazionali:
a è una terza tabella che contiene le righe comuni a entrambi
b è ottenuta dalla seconda tabella che contiene le righe comuni a entrambi
c può essere fatta se le tabelle sono compatibili come per l'unione
d ha come notazione A INTERSEC B

4 Il prodotto di due tabelle relazionali:
a si chiama anche prodotto cartesiano
b è la concatenazione di ogni riga in una tabella con ogni riga della seconda tabella
c è la concatenazione delle righe comuni alle due tabelle
d A (m righe) e B (n righe) è la tabella C (m*n righe)

5 L'operazione di JOIN su due tabelle relazionali:
a combina le operazioni di prodotto e di selezione
b combina le operazioni di intersezione e di selezione
c nel caso in cui un criterio è basato sull'equivalenza dei valori di una colonna si chiama EQUI JOIN
d una JOIN naturale consiste in una EQUI JOIN in cui le colonne ridondanti vengono eliminate

6 L'operazione di JOIN, prevede due varianti:
a LEFT JOIN, che conserva tutte le righe della tabella di sinistra
b LEFT JOIN, che conserva tutte le righe della tabella di destra
c RIGHT JOIN, che conserva tutte righe della tabella di sinistra
d RIGHT JOIN, che conserva tutte righe della tabella di destra

VERO/FALSO

1 Nell'operazione di unione le righe duplicate vengono rimosse. V F
2 Nell'operazione di differenza l'ordine degli operatori è ininfluente. V F
3 Il prodotto di due tabelle è una terza tabella con n • m colonne. V F
4 L'operatore di proiezione recupera un sottoinsieme di colonne da una tabella. V F
5 L'operatore di selezione recupera un sottoinsieme di colonne da una tabella. V F
6 Una EQUI JOIN è in una JOIN naturale in cui le colonne ridondanti vengono eliminate. V F
7 Una JOIN naturale è in una EQUI JOIN in cui le colonne ridondanti vengono eliminate. V F
8 L'operazione di LEFT JOIN conserva tutte le righe della tabella di sinistra. V F

VERIFICA... le competenze

PROBLEMI

Per ogni situazione proposta, progetta lo schema E-R, definisci lo schema logico e popola le tabelle con almeno 5 elementi ciascuna. Su di esse, indica con quale operazione si producono i risultati richiesti.

1 Fotografie

Si vuole rappresentare una base di dati di un archivio fotografico dislocato in varie sedi. Le fotografie sono catalogate in base al soggetto che raffigurano, a colori o in BN. Le foto possono descrivere personaggi, luoghi o oggetti.

a Elencare le foto a colori.
b Elencare le foto a colori della sede di Milano.
c Elencare le foto con la città di Milano.
d Elencare le foto con il papa in visita a Napoli.

2 Vigili del fuoco

I Vigili del fuoco di Milano hanno diverse caserme nel territorio comunale, ciascuna operante in diverse zone della città, individuate da un nome e un codice. In ogni caserma sono presenti diverse squadre di intervento, ciascuna ha un proprio codice (univoco per tutte le squadre) e un proprio simbolo di riconoscimento (un tipo di fiore, non necessariamente univoco),

a Elencare i nomi delle squadre della caserma "Giuseppe Verdi".
b Elencare le zone dove interviene la squadra AK 234.
c Elencare le caserme che hanno una squadra con la margherita.
d Elencare le zone dove interviene le squadra con la rosa.

3 Automobili

Un concessionario plurimandatario vende auto nuove e usate.

a Elencare le macchine nuove disponibili per la vendita.
b Elencare le macchine nuove della Fiat.
c Elencare le macchine usate con meno di 100.000 km.
d Elencare le macchine usate di colore rosso con costo inferiore a 10.000 euro.
e Elencare le macchine vendute a Rossi Mario.

4 Compagnia aerea

Ogni volo di linea di ciascuna compagnia aerea è caratterizzato da un numero, l'orario di partenza e arrivo, la durata. In caso di nebbia o neve il volo viene soppresso.
A ogni viaggio partecipa un pilota e un vicepilota.

a Elencare le compagnie che fanno scalo a Milano.
b Elencare i voli che fanno scalo a Napoli.
c Elencare i voli sospesi per nebbia a Torino.
d Elencare il nome del pilota del volo AZ345.
e Elencare il nome dei vicepiloti che hanno raggiunto Bari con Alitalia.

VERIFICA... le competenze

ESERCIZI

Per le situazioni descritte viene richiesta la progettazione dello schema concettuale e dello schema logico del database. Indica, su di esse, con quale operazione si producono i risultati richiesti.

1 Personaggi dei fumetti

Una casa editrice di riviste a fumetti vuole realizzare un archivio per memorizzare informazioni relative a tutte le storie che ha pubblicato. L'esigenza manifestata dai clienti è di poter conoscere in quali riviste è possibile leggere le avventure dei propri beniamini e, nel caso di suddivisione in più puntate, su quali altri numeri è stata pubblicata una storia. Oltre al titolo, per ogni storia e/o puntata è necessario conoscere il disegnatore che l'ha realizzata e il numero di pagine che la compongono. I protagonisti sono classificati come principali e secondari e viene mantenuta l'eventuale parentela tra gli stessi (per esempio Paperino zio di Qui, Quo e Qua).
Individua le operazioni relazionali per ottenere:
– l'elenco dei personaggi presenti nelle storie di un mese scelto a piacere;
– l'elenco delle storie di un personaggio desiderato pubblicate nell'anno corrente.

2 Appartamenti in affitto

Un'agenzia immobiliare di una località di villeggiatura deve gestire le prenotazioni estive per gli appartamenti per i quali ha il mandato di gestione. Gli appartamenti sono ubicati in palazzi aventi un amministratore, sono identificati in base a un codice e caratterizzati da una metratura, un numero di locali, un piano e un numero di scala. Le locazioni sono per periodi variabili e il costo varia da appartamento ad appartamento, ed è noto il canone mensile richiesto dal proprietario: il contratto viene intestato a un unico cliente, identificato mediante codice fiscale. Per gli affitti di durata inferiore ai tre giorni il prezzo è comunque non inferiore a 200 euro.
Individua le operazioni relazionali per produrre:
– l'elenco degli appartamenti disponibili in immobili con più di 4 piani;
– l'elenco degli appartamenti amministrati da "Rossi Filippo".

3 Musei & opere

Il Ministero dei Beni culturali vuole realizzare un archivio completo per poter censire e offrire un servizio turistico catalogando e classificando tutte le opere presenti nei musei nazionali di un gruppo di artisti. Le opere possono essere dipinti o sculture e per ciascuno di essi viene riportato oltre al titolo, l'autore e la collocazione, la corrente artistica e lo stato di conservazione. Ogni museo ha specifici orari di apertura, costi diversi per i vari periodi dell'anno e le distinte categorie di turisti ed è sotto la responsabilità di un direttore che, in alcuni casi, può anche essere l'unico direttore per tutti i musei della stessa città. La scheda anagrafica degli artisti deve contenere, oltre alla nazionalità e alla data di nascita (ed eventualmente di morte), le correnti artistiche che ha seguito e i periodi di ciascuna di esse. Per i dipinti è importare sapere la tecnica e le dimensioni della tela, mentre per le sculture, oltre alla materia con la quale sono state fatte, interessa conoscerne il peso.
Individua le operazioni relazionali per ottenere:
– l'elenco dei musei che annoverano le opere di un artista prescelto;
– il nome degli artisti e il titolo delle opere conservate nei musei di Firenze.

4 Ricette & calorie

Si vuole realizzare un archivio per la gestione delle ricette domestiche in modo da poter tenere sotto controllo le calorie e preparare menu bilanciati. Ogni piatto viene catalogato, in base al contenuto di calorie, in Classi Caloriche, che possono essere, per esempio, "bassa", "media", "alta" ecc., ed è realizzato seguendo una ricetta che riporta ogni ingrediente con la sua quantità. Ogni ingrediente è, a sua volta, classificato in base alla tipologia alla quale appartiene, come per esempio "pesce", "carne"," verdura" ecc.
Individua le operazioni relazionali per:
– ottenere l'elenco dei piatti ad alto contenuto calorico che contengono zucchero;
– individuare il piatto contenente il maggior numero di calorie.

ALTERNANZA SCUOLA-LAVORO

Scheda progetto n. 1

ARCHIVIO AZIENDE ADERENTI ALL'ALTERNANZA SCUOLA-LAVORO

?

PROBLEMA

Per la realizzazione della alternanza scuola-lavoro, la Camera di Commercio di ogni Provincia, l'Unione Nazionale delle Camere di Commercio e di Infocamere hanno presentato un progetto del Registro Nazionale sull'alternanza scuola-lavoro formato da due sezioni:

- **sezione 1 aperta e consultabile gratuitamente, in cui saranno visibili le imprese e gli enti pubblici e privati disponibili a ospitare i percorsi di alternanza;**
- **sezione 2 riservata, contenente elementi identificativi delle imprese disponibili per l'alternanza scuola-lavoro, come informazioni relative all'attività svolta, al fatturato, al patrimonio netto, al sito internet e al numero di studenti richiesti per indirizzo e specializzazione di studio.**

Sulla base della tua esperienza ti è richiesto di progettare un database che, a partire dai dati presenti nella sezione 1, permetta:

- **l'accoppiamento dei ragazzi alle ditte disponibili;**
- **la stampa automatica di tutti i documenti previsti;**
- **la compilazione giornaliera del diario delle attività.**

Prerequisiti e obiettivi formativi

- **Aver effettuato un periodo di alternanza**
- **Saper analizzare un problema**
- **Saper progettare lo schema di un database**
- **Saper realizzare un database utilizzando un DBMS**
- **Saper estrarre/scambiare dati da due database diversi**
- **Saper integrare dati a documenti elettronici**

Indicazioni per la progettazione

Il problema da risolvere può essere scomposto in fasi:

a acquisizione dei dati presenti nel registro della Camera di Commercio;
b definizione del database;
c definizione della modulistica e produzione dei modelli;
d realizzazione del database in un DBMS;
e importazione dei dati;
f data entry con i dati degli alunni.

La prima fase richiede la connessione a Internet e/o la possibilità di avere a disposizione il tracciato record dei dati presenti nel Registro.

Le fasi b-c sono di progettazione e realizzazione di un database e di definizione dei modelli Word/Excel nei quali inserire successivamente i dati presenti nel database.

ALTERNANZA SCUOLA-LAVORO

Le ultime tre fasi richiedono come prerequisito la conoscenza di un DBMS per realizzare fisicamente le tabelle, inserire i dati e collegarli ai documenti elettronici predisposti: verranno pertanto realizzate dopo lo studio della unità di apprendimento relativa al DBMS.

Traccia per la realizzazione delle prime tre fasi

Per la fase a:

- ricerca sul sito della Camera di Commercio della tua città un referente che possa comunicarti il tracciato record delle tabelle presenti nel Registro delle imprese;
- in alternativa, connettiti al Registro ed effettua un'analisi dei dati individuando il data type di ogni attributo;
- se non fosse disponibile il Registro, predisponi un archivio specifico per memorizzare i dati delle ditte richiesti dalla modulistica in vigore nella tua scuola.

Per la fase b progetta il database definendo le tabelle:

- Aziende
- Settori
- Alunni
- Specializzazioni
- Attività
- Accoppiamenti

Per la fase c predisponi – prendendo spunto da quelli che hai compilato per il tuo periodo di alternanza – quattro moduli Word:

- convenzione;
- piano formativo;
- scheda di valutazione dell'alunno da parte della azienda;
- scheda di valutazione della azienda da parte dell'alunno.

Predisponi inoltre i campi per poi effettuare la connessione dei dati al database.

Preparazione e presentazione dei risultati

La soluzione di questo problema prevede il progetto dello schema logico del database e quindi la presentazione dei risultati del lavoro consiste nella compilazione in formato digitale di tale schema, predisponendo il tracciato record compelto di ogni tabella del database, utilizzando ad esempio Excel per formattare semplicemente i dati.

Analogo discorso per i moduli realizzati per il punto c): dopo aver fatto una stampa degli stessi è opportuno evidenziare i campi nei quali verrà effettuato il mail-merge indicando per ciascuno il nome della tabella dalla quale verranno prelevati.

ALTERNANZA SCUOLA-LAVORO

Proposta operativa n. 1

ANALISI DI MERCATO NUOVE STAMPANTI 3D

PROBLEMA

La divisione Ricerca & Sviluppo necessita di acquistare una stampante 3D per realizzare i prototipi in scala 1:5 dei nuovi modelli della prossima produzione.

Passi operativi

- Richiedi le specifiche al responsabile del progetto.
- Individua le specifiche hardware/software della workstation alla quale verrà collegata.
- Informati sul budget di spesa.
- Utilizzando i motori di ricerca individua i principali portali di e-commerce che hanno il prodotto desiderato.
- Richiedi un preventivo al fornitore di fiducia.
- Compila una scheda comparativa costi/prestazioni da sottoporre al tuo tutor aziendale.
- Progetta un database per la raccolta dei dati.

Proposta operativa n. 2

CONFRONTO TRA LE OFFERTE AZIENDALI DEI GESTORI DI TELEFONIA MOBILE

PROBLEMA

Periodicamente è necessario effettuare una verifica sui piani tariffari delle diverse compagnie telefoniche al fine di migliorare il rapporto costi/benefici e per soddisfare la sempre maggiore richiesta di Gigabyte di navigazione.

Passi operativi

- Individua i principali operatori telefonici.
- Predisponi un modulo dove inserire le proposte in base ai diversi servizi.
- Differenzia le offerte in base alle caratteristiche dei servizi offerti.
- Analizza la possibilità di avere una scontistica in base alla quantità.
- Confronta le offerte con la situazione esistente.
- Compila una scheda comparativa costi/prestazioni da sottoporre al tuo tutor aziendale.
- Progetta un database per la raccolta dei dati.

Key concepts

Designing databases

The characteristics of a DBMS

A DBMS is a (complex!) software system that manages large amounts of persistent and shared data, ensuring their integrity and security by executing coordinated requests, protecting them from malfunctions and implementing an access policy.
A data model is a collection of concepts that are used to describe data, their associations and the constraints that they must respect.
A database consists of a schema that describes its logical structure, and an instance, which consists of its data, stored according to a 3-level organization; a DBMS allows several degrees of physical and logical data independence.

The relational model

The relational model is based on the concept of relation, which broadens the mathematical relation between n domains by associating to each occurrence of a domain a name, called attribute.
The diagram of the relation consists of a name and a set of attributes; the instance of a relation is a set of tuples that associate to each attribute of the schema a value of the corresponding domain.
When no information is provided, a specific value called null is used, which does not belong to any domain. To ensure data integrity, it is possible to set different types of constraints, which determines whether a specific instance is allowed (valid).
The relational constraints include those on domains, on tuples, the constraints on key fields and those related to referential integrity.
Referential integrity constraints are used to set the main correlations between data belonging to different relations.

Designing a database

The design phase of an information system is data-driven and employs a methodology consisting of several stages. Each stage uses a specific model and originates a schema.
During the conceptual design stage, a conceptual model is used that, not taking into consideration the specific aspects of DBMSs and how data are represented from a practical point of view, is a useful compromise between the end result and the reality to be modeled.

Conceptual design

Requirement analysis is fundamental in order to design a database that meets the customer's needs.
Since it is not possible to standardize this stage, one has to rely on good sense and on a series of tools that reduce the risk of blunder, and that can also provide a good documentation of the project.
Several strategies can be applied to the design of the E-R schema; the most widespread, and the one most suitable even in the case of extremely complex projects, is the mixed strategy.

The entity-relationship model

The E-R model is a conceptual model that is very commonly used in database design.
There are several variations of the E-R model, that very often differ from one another only by the type of graphical notation used.
The fundamental concepts of the model are entity, association, attribute, and also identifiers, cardinality, rules and hierarchies.
Usually, the descriptive power of the E-R model is not enough in the design stage; for this reason, it is necessary to write accompanying documentation.

CLIL

Logical design

The purpose of logic design is providing a logic schema that is as close as possible (equivalent) to the starting E-R schema and, at the same time, "efficient."
The alternatives available for the designer to translate the associations between entities depend on the maximum cardinalities at play, which also determine the keys of the relationships that can be obtained. Minimum cardinalities can lead to null values.
As for hierarchies, there are several alternatives available; each of them can originate a translation that may lead to a different end result.

Relational operations

Relational algebra (RA) is a language for databases consisting of a set of operators that can be applied to one or more relations, thus executing a relational operation.
There are six fundamental operators: selection, projection, intersection, natural JOIN, union, and difference. Other operators (for example, RIGHT-JOIN and LEFT-JOIN) are based on the fundamental ones.
Broadly speaking, a query on a DB can be represented in RA by means of several expressions, equivalent as far as the end result is concerned, but not necessarily equally efficient.

Quiz

1 What is a database?
- **a** A collection of shared and logically related data, whose purpose is satisfying the informational needs of a specific organization.
- **b** A collection of Java scripts
- **c** A DBMS manager software
- **d** An interpreter of C++ scripts

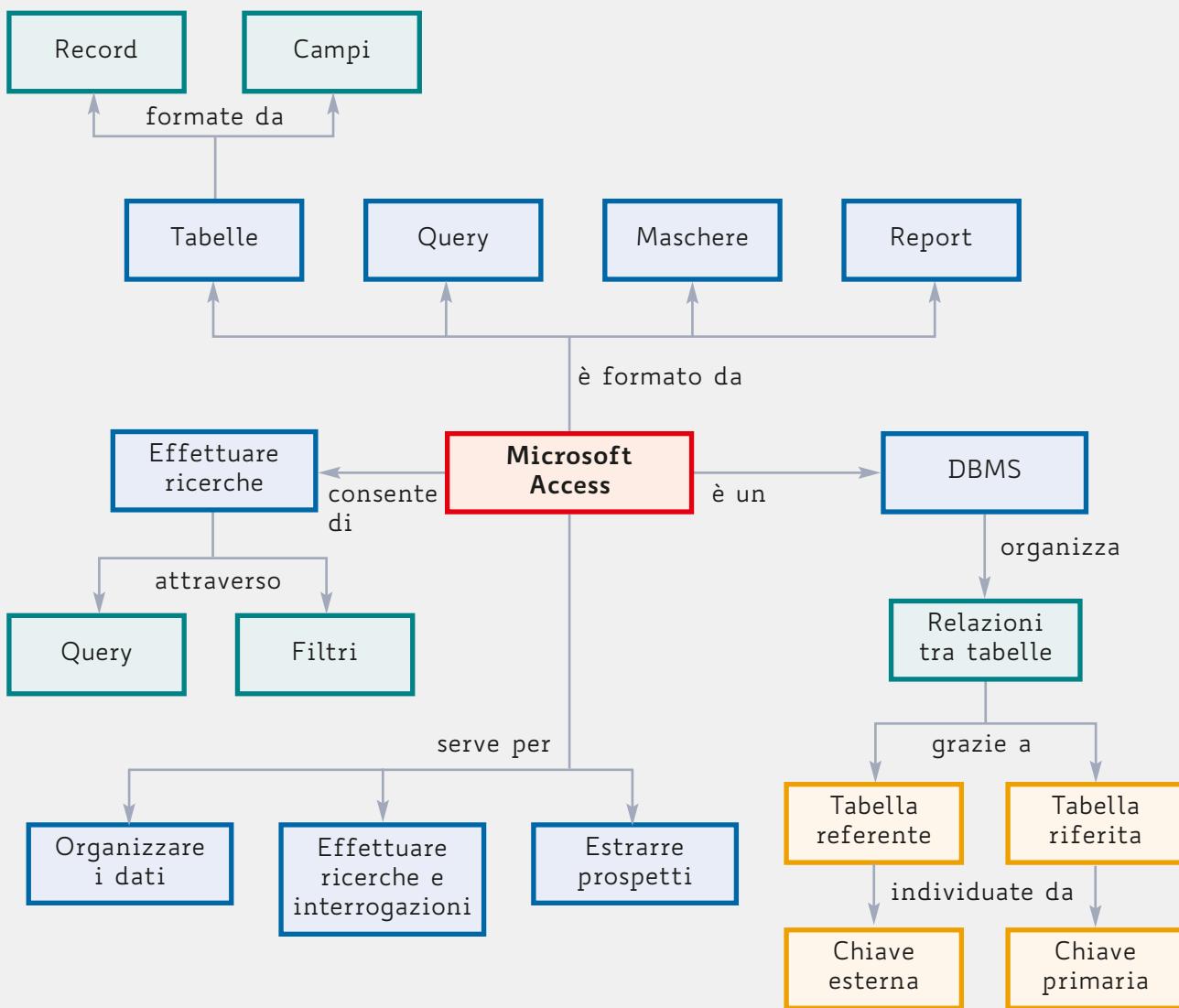
2 What does a DBMS do?
- **a** It manages the data contained within a database and controls access to them
- **b** It is used to display HTML pages
- **c** It manages the files saved on the desktop of a computer
- **d** It interprets PHP scripts

3 Which are the models a database can be based on?
- **a** There are no models for a database
- **b** The E-R model
- **c** DDL, DML, QL
- **d** Hierarchic, reticular, relational, and object-based model

4 What are OODBs (Object-Oriented DataBases)?
- **a** They are DBMSs that use the relational model to manage objects
- **b** They are database-oriented objects
- **c** They are databases that manage objects and classes
- **d** They are the fundamental elements of the SQL language

5 Which one among the following is not a relational operation?
- **a** Projection
- **b** Conjunction
- **c** Selection
- **d** Union

6 What is the purpose of the E-R (entity–relationship) model?
- **a** It is the real tool used to analyze the management performance of data
- **b** It is the tool used to analyze the characteristics of a certain reality independently from the events happening in it
- **c** It is mostly used to identify the relationships between entities and between attributes of the same entity, and transmit the result
- **d** It is a model that summarizes the procedures to follow to determine entities and relationships.

Database Management System (DBMS)

CONOSCENZE

- Riconoscere il ruolo dei DBMS
- Individuare gli elementi che costituiscono le basi di dati
- Riconoscere la struttura di una tabella
- Individuare il ruolo dei diversi tipi di query

COMPETENZE

- Definire la struttura delle tabelle
- Applicare le interrogazioni di selezione e di raggruppamento
- Applicare gli operatori di aggregazione
- Definire report personalizzati
- Applicare le procedure macro alle maschere

ABILITÀ

- Rappresentare i dati mediante tabelle
- Raffigurare i dati con maschere personalizzate
- Estrarre dati mediante prospetti
- Creare automatismi con le macro

Il file relativi a esempi, esercizi e a eventuali soluzioni di questa unità sono contenuti nel CD-ROM e sono anche scaricabili dal sito hoepliscuola.it

Introduzione ad Access

IN QUESTA LEZIONE IMPAREREMO...

- a riconoscere gli oggetti del DBMS Access
- a utilizzare Access per creare tabelle
- a individuare i tipi di campo disponibili
- a relazionare le tabelle

MAPPA CONCETTUALE

Record
Campi
formate da
Tabelle
Query
Maschere
Report
è formato da
Effettuare ricerche
consente di
Microsoft Access
è un
DBMS
organizza
attraverso
Query
Filtri
Relazioni tra tabelle
grazie a
serve per
Tabella referente
Tabella riferita
Organizzare i dati
Effettuare ricerche e interrogazioni
Estrarre prospetti
individuate da
Chiave esterna
Chiave primaria

Basi di dati

Una **base di dati** è composta fondamentalmente da tabelle, campi e record. Il DBMS, come per esempio Access, è il software che consente di creare nuove basi di dati e di effettuare ricerche su di esso.

Base di dati
Una base di dati o database è un insieme di moltissime informazioni memorizzate per un lungo periodo di tempo, gestite da un software chiamato DBMS (Data Base Management System).

Ridondanza
Il termine ridondanza indica la presenza di dati ripetuti all'interno dello stesso archivio.

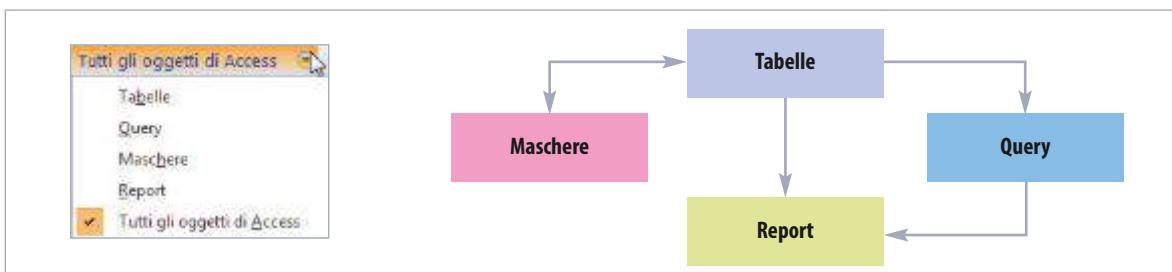
Le funzioni principali di un DBMS consistono essenzialmente nel:

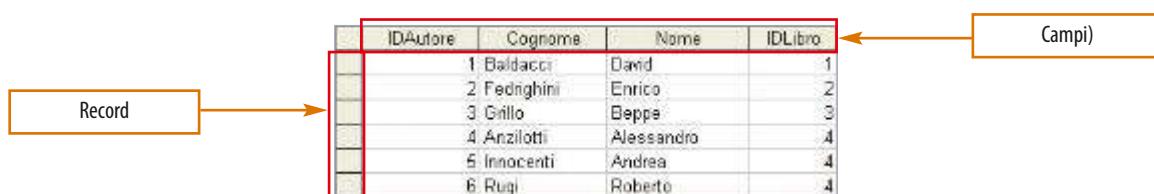
- fornire all'utente strumenti per la creazione di basi di dati e specificare una struttura logica per i dati chiamata schema o struttura;
- dare agli utenti la possibilità di effettuare ricerche nella base di dati (interrogazioni);
- permettere la memorizzazione di grandi quantità di dati eterogenei (big data) per un lungo periodo di tempo, preservandoli da incidenti e usi non autorizzati;
- controllare gli accessi temporanei da parte di più utenti.

I DBMS sono software molto diffusi in tutte quelle situazioni nelle quali è necessario memorizzare ingenti quantità di dati (biblioteche, elenchi telefonici online, ospedali, banche ecc.).

La sicurezza

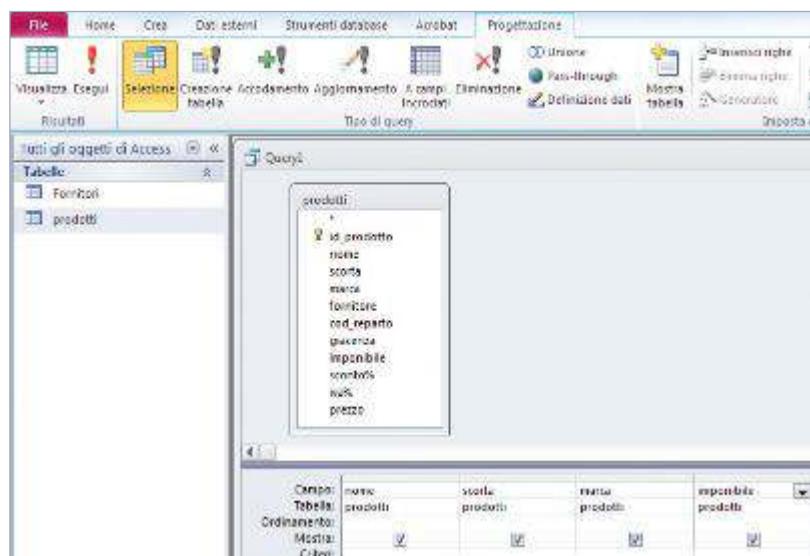
Per garantire la sicurezza dei dati gestiti da un DBMS esistono diverse tecniche; di seguito citiamo quelle principali.

- Cifratura o criptaggio. I dati possono essere interpretati solo attraverso l'uso di una chiave di lettura o di modifica segreta.
- Partizionamento. La suddivisione del database avviene in sezioni aventi ciascuna una sensibilità diversa. Questa tecnica aumenta la **ridondanza** delle informazioni, ma garantisce una maggiore integrità in quanto consente l'accesso a una sola sezione del database.
- Lock di integrità. I dati vengono marcati attraverso etichette che definiscono la sensibilità e i permessi di accesso, in modo da proteggere le informazioni da modifiche accidentali o da accessi non autorizzati.
- Front end fidato. Viene anche chiamata tecnica multilivello o guardia. L'utente deve passare attraverso due "filtri" per accedere ai dati, il primo chiamato front end e il secondo fidato.
- Finestre e viste. L'utente può avere accesso solo a una finestra, che è un sottoinsieme dei dati. Questo impedisce all'utente di avere accesso a tutti i dati presenti nel database.

Gli oggetti di Access

Ciascun database di Access viene memorizzato all'interno di un file con estensione .accdb.

Le estensioni dei file di Access sono .mdb fino alla versione 2003 e .accdb dalla versione 2007 in poi.

Ogni database di Access è formato dagli oggetti seguenti.

Come possiamo notare dalla figura di destra, la relazione tra i diversi elementi prevede che a partire da una tabella possano essere generate Maschere, Query e Report, mentre da una Query possano essere generati Report. Da questo schema comprendiamo la centralità del ruolo delle Tabelle, che rappresentano la parte principale del database; in esse risiedono i dati elementari sui quali opera il DBMS.
Un database di Access può contenere diverse tabelle, ciascuna delle quali possiede una struttura ed è formata da righe chiamate record e da colonne chiamate campi.

IDAutore	Cognome	Nome	IDLibro
1	Baldacci	David	1
2	Fedrighini	Enrico	2
3	Grillo	Beppe	3
4	Anzilotti	Alessandro	4
5	Innocenti	Andrea	4
6	Rugi	Roberto	4

Durante la fase di creazione di una tabella dobbiamo decidere il nome dei vari campi e il loro tipo e formato. Dalle tabelle possono essere create Query, Maschere e Report.

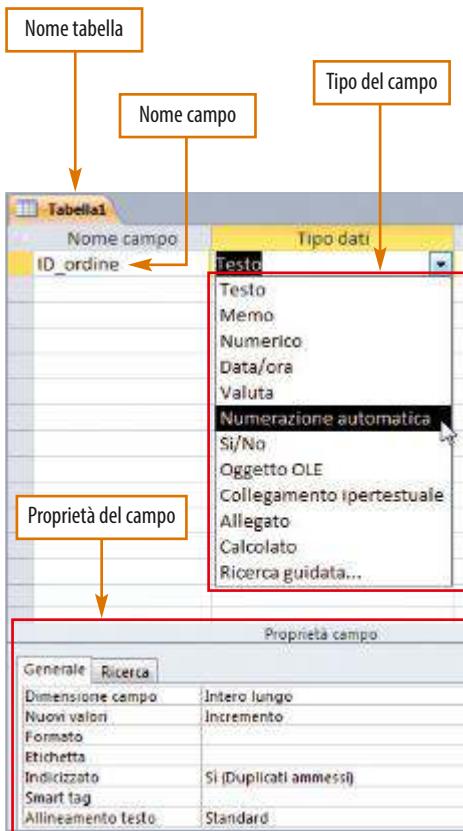
Immaginiamo di voler creare una tabella per archiviare i nostri libri di casa.
Possiamo individuare i campi per queste informazioni: autore, titolo, casa editrice, prezzo ecc.
Per registrare effettivamente i dati di ciascun libro useremo invece i record.

Query

Mediante le query (in italiano: "richieste") possiamo interrogare il database, cioè effettuare ricerche all'interno di una o più tabelle attraverso criteri specifici. Una query, sostanzialmente, estrae i dati perché vengano visualizzati, stampati, cancellati o inseriti in una nuova tabella. Dalle query possono essere creati sia maschere sia report.

L'immagine a lato mostra la finestra di QBE (*Query By Example*), che consente la progettazione di una query di selezione.

Maschere

Le maschere consentono di visualizzare e modificare i dati presenti nelle tabelle o nelle query in modo agevole, rendendoli esteticamente più gradevoli, come possiamo vedere nell'esempio della figura a lato.

Report

I report sono strumenti che permettono di creare stampe di qualità, chiamate “prospetti personalizzati”. Tramite questo strumento si possono stampare elenchi, statistiche, grafici, raggruppare i dati con totali parziali scegliendo stili, caratteri, impaginazione e così via.

Creare una nuova tabella

Vediamo come creare una tabella adatta a contenere i dati di una rubrica telefonica attraverso la procedura che segue.

1. Nella finestra che appare all’apertura di Access, selezioniamo la voce Nuovo e quindi Database Vuoto, se non desideriamo utilizzare un modello predefinito; utilizziamo la voce Database Vuoto per crearlo da zero.
2. A questo punto, viene richiesto il nome da assegnare al file del database, mentre l’estensione .accdb viene assegnata automaticamente dal programma.
3. Per creare una nuova tabella utilizziamo il pulsante Struttura tabella, presente nel gruppo Tabelle della scheda Crea.
4. Appare la finestra che mostra la struttura della tabella da creare. Come possiamo notare, vi sono due colonne, una per il nome del campo (Nome campo) e un’altra, accanto, per il tipo di campo (Tipo dati): digitiamo ID_ordine e assegniamo a esso il tipo Numerazione automatica.
5. Aggiungiamo anche il campo Data_ordine, assegnando a esso il tipo Data/ora.

Per questo tipo di campo modifichiamo la Proprietà Formato, scegliendo Data in cifre.

6. Dopo aver aggiunto il campo Importo (di tipo Valuta), aggiungiamo il campo Cliente (di tipo Testo) e modifichiamo la proprietà Dimensione campo (50 caratteri). Aggiungiamo poi il campo Evaso, assegnandogli il tipo Sì/No.
7. A questo punto, per salvare la tabella è sufficiente fare clic sul pulsante di chiusura posto sulla barra del titolo e assegnarle il nome Ordinativi.
8. Il messaggio informativo che appare subito dopo indica che non abbiamo creato alcuna **chiave primaria**. Nella finestra facciamo clic su No, per evitare che venga aggiunto un campo di tipo Chiave primaria.

Chiave primaria
Con l'espressione "chiave primaria" (primary key) si indica un campo della struttura di una tabella i cui valori individuano univocamente ciascun record della tabella, in modo che non possano esistere due o più record con lo stesso valore di campo chiave primaria. Alcuni esempi sono il codice fiscale, il numero di matricola di uno studente, oppure il numero IBAN del conto corrente.

Modificare la struttura della tabella

La struttura di una tabella può essere modificata per diversi motivi, per esempio per aggiungere un nuovo campo, eliminarlo, spostarlo, oppure modificarne le proprietà. La procedura che segue descrive come modificare la struttura della tabella Ordinativi, aggiungendo a essa alcuni campi.

1. Selezioniamo la tabella creata in precedenza (Ordinativi). Facciamo poi clic col tasto destro del mouse sull'icona della tabella e selezioniamo la voce Visualizzazione Struttura.
2. Iniziamo con l'assegnare la chiave primaria al campo ID_ordine: per fare questo, facciamo clic sul nome del campo con il tasto destro del mouse, quindi scegliamo Chiave primaria.
3. Possiamo anche inserire o cancellare le righe della struttura della tabella usando il menu contestuale che appare facendo clic con il tasto destro sul mattoncino posto accanto al nome del campo.

PER SAPERNE DI PIÙ

CAMPO A TENDINA

Sempre con riferimento alla tabella Ordinativi, illustriamo la procedura che ci consente di poter selezionare il nominativo del cliente da un **campo a tendina**.

Campo a tendina
Viene anche chiamato campo a casella combinata. Il campo a tendina consente all'utente, durante l'inserimento dei dati nella tabella, di selezionare il dato da inserire da un elenco.

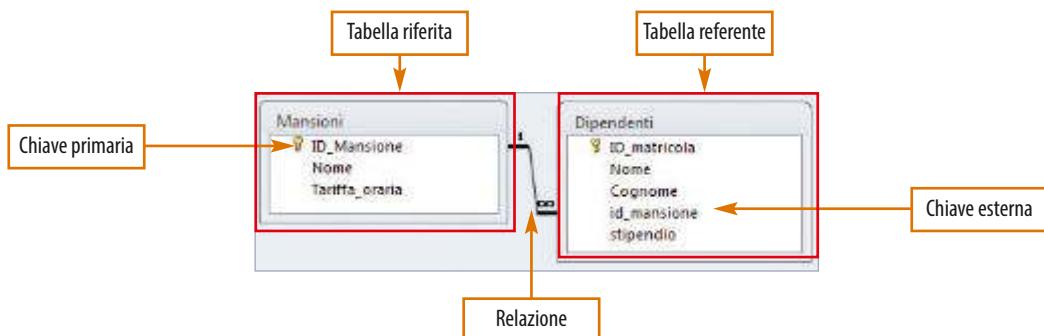
1 Posizioniamoci nella sezione Proprietà del campo e scegliamo la scheda Ricerca del campo Cliente.
2 Selezioniamo la voce Visualizza controllo e facciamo clic su Casella combinata.

3 Selezioniamo quindi la voce Tipo origine riga e facciamo clic su Elenco valori; scegliamo poi la voce Origine riga e scriviamo nella casella a fianco l'elenco che vogliamo far apparire nella colonna del campo durante la fase di inserimento record. L'elenco è formato da stringhe di testo racchiuse tra doppi apici e separate dal punto e virgola.
Inseriamo alcuni nominativi di clienti.

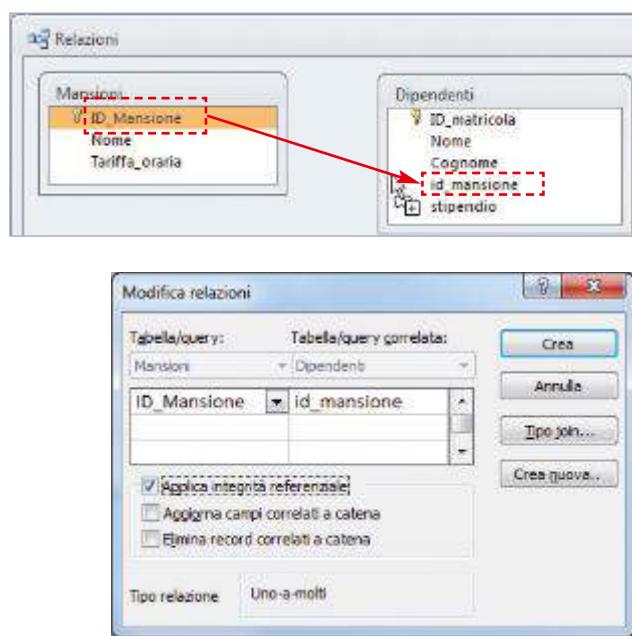
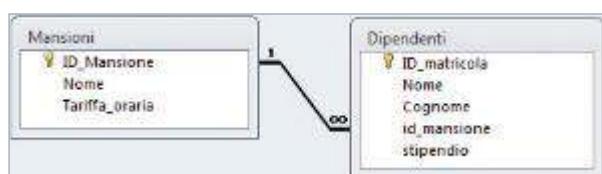
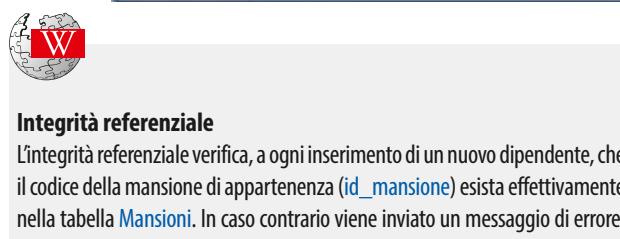
4 Dopo aver salvato la tabella, apriamola facendo doppio clic su di essa: possiamo notare che, posizionandoci sul campo del Cliente, appare il menu a tendina dal quale possiamo selezionare il nominativo interessato.

Le relazioni

Esistono alcuni vincoli relativi al legame esistente tra più tabelle. Access è infatti un database relazionale: la parola "relazionale" deriva dal fatto che in un database di Access vi possono essere più tabelle, collegate tra loro da relazioni.
Per creare i database più complessi, gli analisti (coloro che progettano le basi di dati) creano moltissime tabelle che nascono da un procedimento matematico chiamato normalizzazione.

Per capire in che cosa consiste, facciamo un piccolo esempio.
Immaginiamo di voler creare una tabella che contenga i dati di una rubrica telefonica, come nell'esempio dei paragrafi precedenti. Non esistono problemi fintanto che ciascun elemento della tabella possiede un solo numero telefonico, ma se prendiamo in esame il caso in cui a ciascun nominativo possano essere associati da nessuno a dieci diversi recapiti telefonici, ci rendiamo facilmente conto che è inutile inserire dieci campi per ogni nominativo. Se trasferissimo i recapiti telefonici in un'altra tabella avremmo risolto il problema, tuttavia perderemmo il legame tra il nominativo e il relativo recapito.

Per risolvere questo problema, basta mantenere, nella tabella dei recapiti, un campo (detto chiave esterna) che ci permette di sapere qual è il record della tabella della rubrica al quale fa riferimento quel particolare recapito.

Relazione uno-a-molti tra tabelle (Mansioni e Dipendenti)

Supponiamo di avere due tabelle, una chiamata Mansioni, e una seconda chiamata Dipendenti. La tabella Mansioni contiene le mansioni identificate dal codice presente nel campo ID_Mansione, che ha valore di chiave primaria, mentre la tabella Dipendenti contiene i nominativi dei dipendenti di un'azienda: il campo id_mansione rappresenta la chiave esterna che fa riferimento al campo ID_Mansione della tabella Mansioni.

Le due tabelle (Mansioni e Dipendenti) vengono rispettivamente indicate come tabella riferita e tabella referente.

Facciamo notare che il campo chiave primaria è stato indicato con "ID" scritto in maiuscolo, mentre il campo chiave esterna con "id" scritto in minuscolo. Si tratta unicamente di una convenzione, che ci aiuta nell'identificare la tabella riferita rispetto alla tabella referente.

Assegnando la chiave primaria al campo ID_Mansione otteniamo che il codice identificativo di ogni singola mansione (per esempio dirigente, impiegato, operaio ecc.) potrà essere presente nella tabella Mansioni una sola volta, mentre il valore del suo corrispettivo nella tabella Dipendenti potrà comparire varie volte, tante quanti sono i dipendenti che hanno quella particolare mansione all'interno dell'azienda. Ecco spiegato il significato dei simboli che appaiono nella schermata sopra riportata: il significato di 1 e ∞ è uno (1) a molti (infinito).

La procedura che descriviamo di seguito illustra come creare una relazione uno-a-molti tra due tabelle, in questo caso la tabella Mansioni con la tabella Dipendenti.

1. Per creare una nuova relazione tra due tabelle dobbiamo fare clic sul pulsante Relazioni posto nella scheda Strumenti database.
2. Bisogna quindi collocare le due tabelle all'interno del riquadro Relazioni. Per fare questo, facciamo clic sul pulsante Aggiungi: selezioniamo la prima tabella (Mansioni) dall'elenco, e ripetiamo l'operazione per la seconda tabella (Dipendenti).
3. Trasciniamo adesso con il mouse il campo chiave primaria della tabella Mansioni (ID_Mansione) sul campo chiave esterna della tabella Dipendenti (id_mansione).
4. Quando rilasciamo il pulsante, appare la finestra mostrata a lato nella quale dobbiamo spuntare la voce **Applica integrità referenziale**.
5. Dopo aver fatto clic su Crea otteniamo la seguente relazione.

Integrità referenziale
L'integrità referenziale verifica, a ogni inserimento di un nuovo dipendente, che il codice della mansione di appartenenza (id_mansione) esista effettivamente nella tabella Mansioni. In caso contrario viene inviato un messaggio di errore.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Quale tra le seguenti non è una tipica funzione di un DBMS?
- **a** Fornire uno strumento sicuro per il controllo degli accessi da parte di più utenti
- **b** Fornire la possibilità di effettuare ricerche dei dati
- **c** Fornire uno strumento per garantire l'accesso al database da remoto
- **d** Fornire uno strumento per definire la struttura dei dati

2 Quale tra le seguenti non è una tecnica di un DBMS utile a garantire la sicurezza dei dati?
- **a** Il lock di integrità
- **b** Le viste e le finestre
- **c** Il partizionamento
- **d** Il report

3 Un campo di tipo data può essere utilizzato come chiave primaria di una tabella contenente fatture commerciali?
- **a** Sì, sempre
- **b** No: in questo caso avremmo più fatture nella stessa data
- **c** Sì, solo se la data è in formato commerciale (gg/mm/aa)
- **d** No: in questo caso avremmo solo una fattura per ciascuna data

4 Quale tecnica tra le seguenti garantisce la sicurezza dei dati? *(2 risposte)*
- **a** Ridondanza
- **b** Finestre e viste
- **c** User Friendly
- **d** Cifratura e criptaggio

5 Quali tra i seguenti oggetti di Access permette di ottenere stampe personalizzate a partire da una query?
- **a** Query
- **b** Maschere
- **c** Report
- **d** Tabelle

6 Nella definizione della struttura di una tabella quale tipo di campo tra i seguenti è il più adatto a contenere un numero di telefono?
- **a** Valuta
- **b** Testo
- **c** Numerico
- **d** Data ora

7 Quale caratteristica deve possedere il campo oggetto di relazione, presente nella tabella referente?
- **a** Non deve avere nessuna caratteristica particolare
- **b** Deve essere di tipo a Numerazione Automatica
- **c** Deve essere una Chiave primaria
- **d** Deve essere una Chiave esterna

8 Le colonne di una tabella rappresentano:
- **a** i record
- **b** i campi
- **c** le query
- **d** le chiavi primarie

9 Da quale oggetto di Access può essere creata una query? *(2 risposte)*
- **a** Query
- **b** Tabelle
- **c** Report
- **d** Maschere

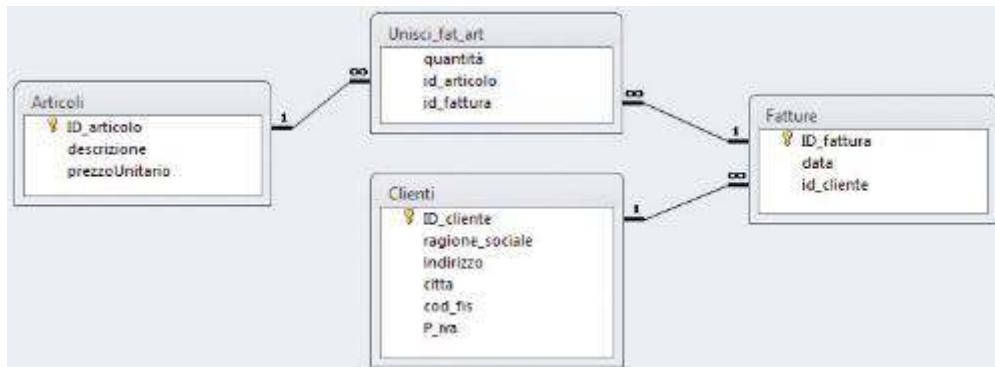
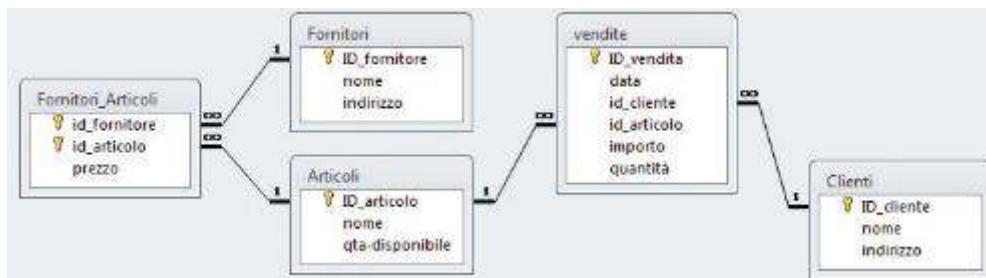
10 Quale tra le seguenti affermazioni indica correttamente qual è il ruolo dell'integrità referenziale?
- **a** Verifica che i valori contenuti dei campi siano tutti diversi
- **b** Controlla che ad ogni record inserito nella tabella referente esista una chiave primaria nella tabella riferita
- **c** Controlla che ad ogni record inserito nella tabella referente esista un record nella corrispettiva tabella riferita
- **d** Seleziona i campi attraverso i criteri

11 Quale caratteristica deve possedere il campo oggetto di relazione, presente nella tabella riferita?
- **a** Deve essere una chiave primaria
- **b** Deve essere una chiave esterna
- **c** Non deve avere nessuna caratteristica particolare
- **d** Deve essere di tipo Numerico

12 Quale tra i seguenti non è un tipico esempio di utilizzo di DBMS?
- **a** Gestione di un magazzino
- **b** Gestione dei clienti
- **c** Gestione della rete informatica di un'azienda
- **d** Gestione delle cartelle cliniche dei pazienti di un ospedale

13 Una tabella è organizzata secondo uno schema chiamato:
- **a** congiunzione
- **b** relazione
- **c** record
- **d** struttura

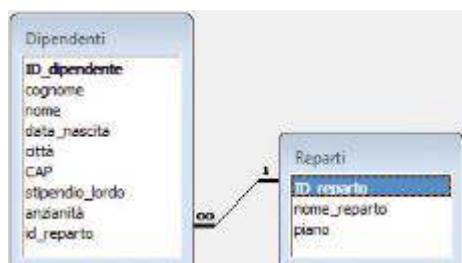
VERIFICA... le competenze

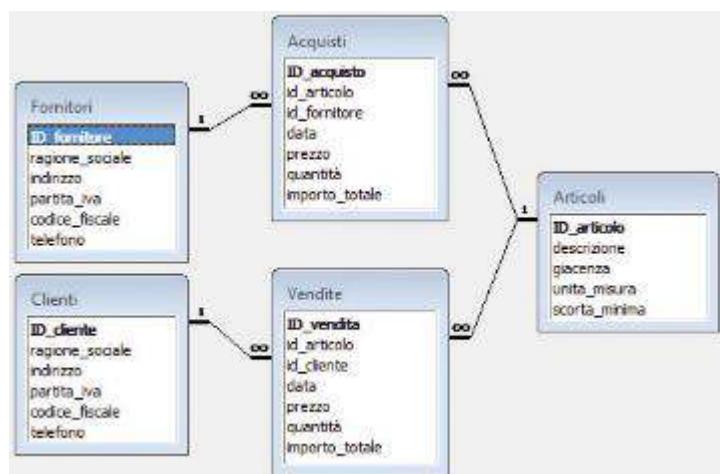
AREA DIGITALE

ESERCIZI

1 Apri il file Fatturazione.accdb
Utilizzando gli strumenti studiati in questa lezione, metti in relazione le tabelle come illustrato dalla immagine seguente.

2 Apri il file Clienti_fornitori.accdb
Utilizzando gli strumenti studiati in questa lezione, metti in relazione le tabelle come illustrato dalla immagine seguente.

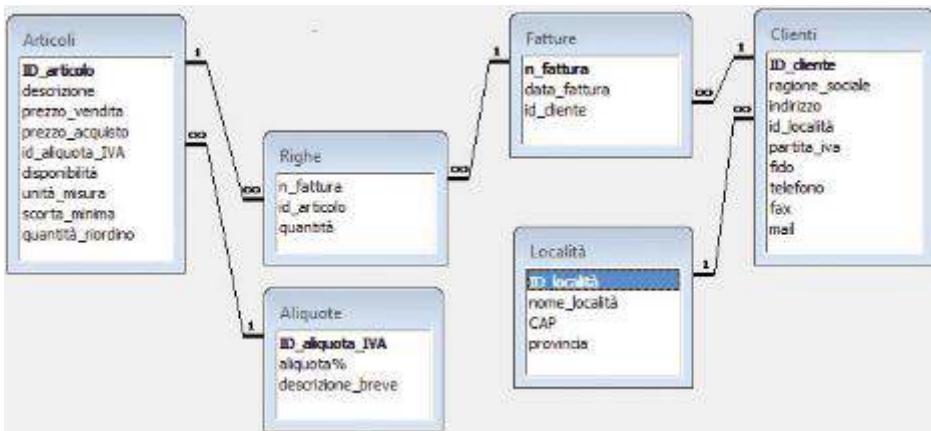
3 Apri il file Negozi.accdb
Utilizzando gli strumenti studiati in questa lezione, metti in relazione le tabelle come illustrato dalla immagine seguente.

VERIFICA... le competenze

4 Crea il file Dipendenti.accdb

– Crea una tabella Dipendenti con i campi di seguito indicati, assegnando il formato corretto:
  ID_dipendente
  cognome
  nome
  data_nascita
  CAP
  città
  stipendio_lordo
  anzianità
  id_reparto

– Crea una tabella Reparti con i campi di seguito indicati, assegnando il formato corretto:
  ID_reparto
  nome_reparto
  id_responsabile

– Metti in relazione le due tabelle come riportato a lato.

5 Crea il file Maga.accdb

Crea le tabelle Articoli, Clienti, Fornitori, Acquisti e Vendite e le relative relazioni come indicato nella figura seguente:

– Crea una query che calcoli la giacenza degli articoli sottraendo alla quantità dei pezzi acquistati quelli venduti.
– Crea una query che calcoli la media degli articoli acquistati per ciascun fornitore.
– Crea una query che calcoli il prezzo di vendita minimo e massimo per ogni articolo.
– Crea una query che mostri i prodotti che hanno la giacenza uguale o inferiore alla scorta minima.
– Confronta la tua soluzione con quella presente nel file Maga_solux.accdb.

VERIFICA... le competenze

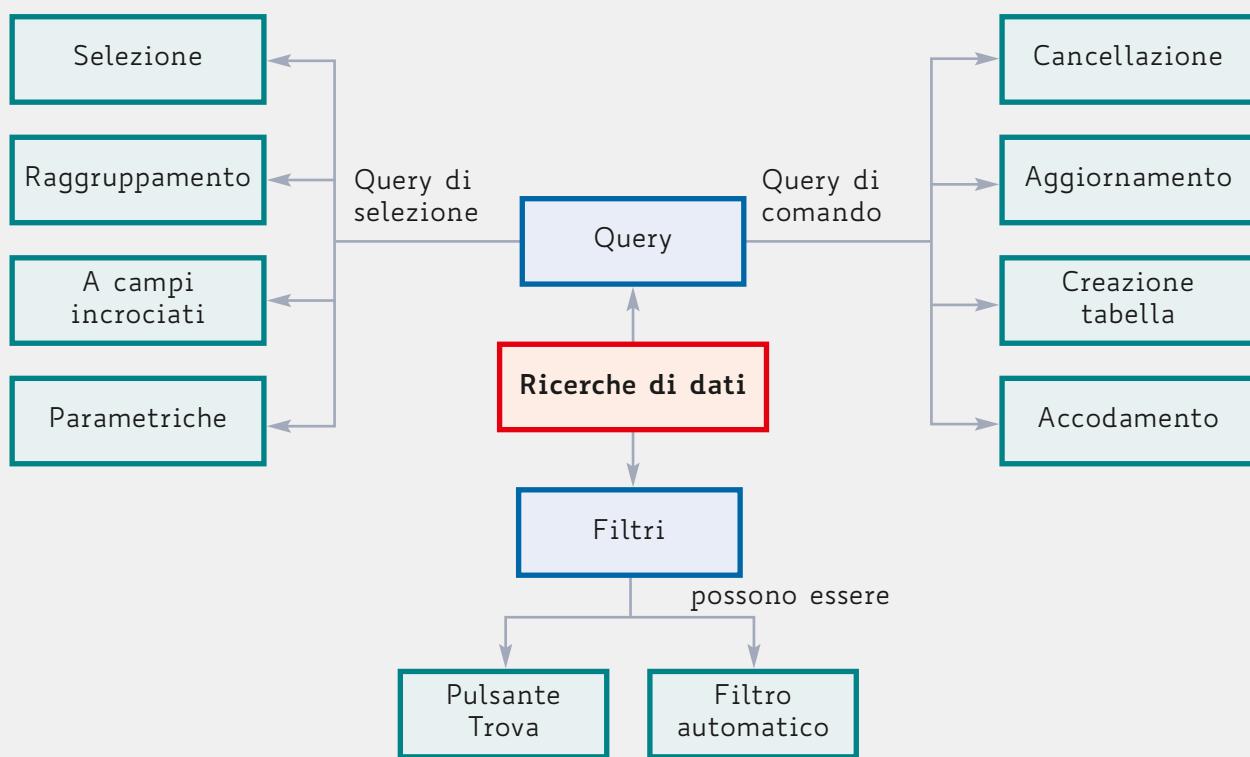
6 Crea il file GeVe.accdb

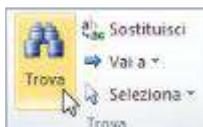
Crea le tabelle Articoli, Clienti, Fatture, Righe, Aliquote, Località e le relative relazioni, come illustrato nella figura seguente.

- Crea una query che calcoli il prezzo medio di vendita degli articoli acquistati dal cliente Corti utilizzando come criterio "like corti*".
- Crea una query che mostri tutte le fatture emesse al cliente di codice 1
- Crea una query che mostri tutte le fatture emesse nel mese di febbraio.
- Crea una query che calcoli il prezzo di acquisto massimo e minimo degli articoli con ancora disponibilità.
- Confronta la tua soluzione con quella presente nel file GeVe_solux.accdb

7 Apri il file Rubrica.accdb

Apri la tabella Rubrica in Visualizzazione struttura ed effettua su di essa le seguenti operazioni:
- aggiungi un campo di nome Città di tipo Testo;
- aggiungi due campi di nome Altezza e Peso, entrambi di tipo Numerico;
- assegna la proprietà Dimensione Campo di tipo Precisione Doppia a entrambi i campi numerici e il formato Numero generico;
- aggiungi un campo di nome codice_IBAN di tipo Testo di dimensione 27;
- aggiungi un campo di nome saldoContoCorrente di tipo Numerico e assegna a esso una Dimensione Precisione Doppia, e il Formato Valuta.

8 Apri il file Temperature.accdb
- Aggiungi la chiave primaria al campo ID_temperatura della tabella Temperature.
- Aggiungi la chiave primaria al campo ID_capoluogo della tabella Capoluoghi.
- Aggiungi la chiave primaria al campo ID_giorno della tabella Giorni.
- Crea una relazione uno a molti con integrità referenziale tra i campi ID_capoluogo della tabella Capoluoghi e id_capoluoghi della tabella Temperature.
- Crea una relazione uno a molti con integrità referenziale tra i campi ID_giorno della tabella Giorni e id_giorni della tabella Temperature.

I filtri e le query

IN QUESTA LEZIONE IMPAREREMO...

- a effettuare una selezione dei record mediante i filtri
- a effettuare ricerche nelle tabelle
- a interrogare il database mediante le query

MAPPA CONCETTUALE

didattica inclusiva

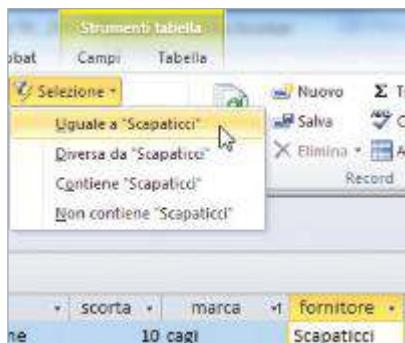
Ricerche di dati → Query; Ricerche di dati → Filtri

Query di selezione: Selezione, Raggruppamento, A campi incrociati, Parametriche

Query di comando: Cancellazione, Aggiornamento, Creazione tabella, Accodamento

Filtri possono essere: Pulsante Trova, Filtro automatico

I filtri

Per ricercare dati in una tabella possiamo utilizzare due diverse tecniche, la ricerca e il filtro. La ricerca avviene posizionandosi dapprima sul campo della tabella nel quale detta ricerca va effettuata e poi facendo clic sul pulsante Trova, presente nel gruppo Trova della scheda Home.

Per effettuare una ricerca basata su una parte di testo, inseriamo alla fine del testo il carattere asterisco ("*"). Digitando per esempio "tab*", vengono cercati tutti i record che iniziano con il testo indicato, per esempio "tablet", "tabella", "tabulato" ecc. Se invece anteponiamo il carattere asterisco al testo, per esempio "*pad", viene effettuata la ricerca dei record che terminano con il testo indicato, per esempio "ipad", "superpad", "musicpad" ecc.

Vediamo adesso come effettuare una ricerca tramite filtro, applicando un filtro automatico per ricercare tutti i prodotti di un dato fornitore.
L'uso del filtro, a differenza della ricerca con il pulsante Trova, consente di ottenere l'elenco dei soli record che soddisfano la condizione di ricerca impostata.
La procedura che segue illustra come filtrare i record della tabella Prodotti del database Supermarket.accdb.

1. Apriamo la tabella Prodotti e posizioniamo il cursore sul campo Fornitore selezionando il fornitore da ricercare, in questo caso "Scapaticci".

2. Nella scheda Home, facciamo clic sul pulsante Selezione, quindi sulla voce che indica la condizione di filtro, in questo caso tutti i record uguali al campo selezionato (Uguale a Scapaticci).
3. Come possiamo notare, appaiono in elenco solo i record che soddisfano la condizione di avere come fornitore "Scapaticci".

Prodotti

id_prodotto	nome	scorta	marca	fornitore	id_reparto
1	calze lunghe cotone	10	cagi	Scapaticci	7
32	calze corte cotone	5	Yanaday	Scapaticci	7
33	salvascarpe donna	20	Yanaday	Scapaticci	7
34	maglia cotone bambino	4	cagi	Scapaticci	7
35	termica invernale	5	cagi	Scapaticci	7
0		20	Happiness	Scapaticci	7

Per disattivare il filtro e continuare a visualizzare l'elenco di record originale, bisogna fare clic sul pulsante Attiva/Disattiva filtro della scheda Home.

Le query

Query
Il termine, che deriva dalla lingua inglese, significa "domanda", "quesito", "interrogazione", indica l'obiettivo dell'oggetto di Access in questione, che è proprio quello di ricercare informazioni nel database.

Le **query** vengono in generale utilizzate per effettuare operazioni sui dati presenti nelle tabelle, effettuando una richiesta sulla base di determinate condizioni (criteri di ricerca).
La struttura di una query è costituita da una griglia, chiamata QBE (*Query By Example*), nella quale si inseriscono per trascinamento i campi e i relativi criteri di ricerca.
Aprire o eseguire una query, invece, significa visualizzarne i risultati di ricerca.

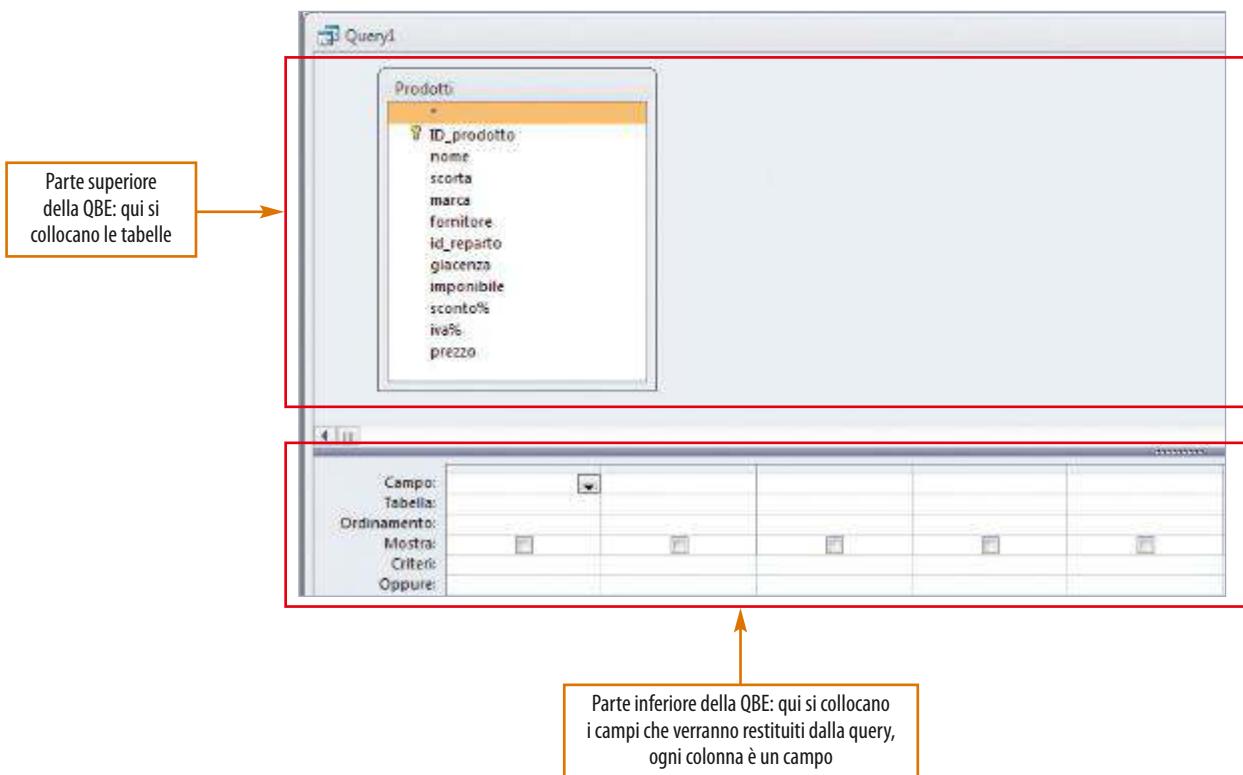
Bisogna distinguere tra query di comando, che servono per cancellare, aggiornare, inserire record in una nuova tabella o in una tabella esistente e query di selezione, che invece consentono di ricercare i dati presenti nelle tabelle.
Le icone delle query di comando sono affiancate da un punto esclamativo (!), per evidenziare il fatto che modificano il contenuto delle tabelle.

Nella tabella che segue, riportiamo tutti i tipi di query che possono essere create in Access.

Query di selezione				Query di comando			
Selezione	Totali	A campi incrociati		Aggiornamento	Creazione tabella	Eliminazione	Accodamento
query di selezione	query di raggruppamento	query a campi incrociati	query parametriche	query di aggiornamento	query di creazione tabella	query di eliminazione	query di accodamento

Query di selezione dati

Le query di selezione si usano per effettuare una ricerca tra i dati delle tabelle. La ricerca può comportare la selezione dei record (attraverso i criteri) e dei campi (attraverso la selezione delle colonne nella QBE).
La procedura che segue illustra come realizzare una query per la ricerca dei prodotti di marca "Findus".

1. Nella scheda Crea, gruppo Query, facciamo clic sul pulsante Struttura query.
2. Facciamo doppio clic sulla tabella sulla quale effettuare la ricerca, in questo caso Prodotti, poi facciamo click su Chiudi.

> Se ci fossero tabelle poste in relazione tra loro, nella query vanno aggiunte solo le tabelle strettamente necessarie. In questo caso, vogliamo ricercare i prodotti di una data marca, pertanto dobbiamo inserire la sola tabella Prodotti.

3. La finestra che appare prende il nome di QBE. Ciascuna colonna rappresenta la colonna della query da creare. Possiamo scegliere il campo da inserire trascinandolo dall'elenco dei campi fino alla colonna prescelta.

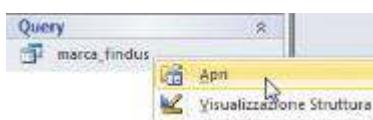
Il riquadro in alto visualizza la tabella o le tabelle utilizzate nella query.

Il riquadro in basso, invece, visualizza i campi nel seguente modo:
- Campo: nome del campo (la freccia consente di scegliere altri campi);
- Tabella: tabella da cui vengono prelevati i dati relativi al campo;
- Ordinamento: consente di ordinare i risultati della query in modo crescente o decrescente;
- Mostra: consente di visualizzare o nascondere un campo nel foglio di risposta (per esempio, può essere utilizzato un campo nascosto per ordinare i dati);
- Criteri: consente di indicare i criteri per la selezione dei dati;
- Oppure: consente di aggiungere criteri.

PER SAPERNE DI PIÙ

COSA INSERIRE NEI CRITERI

Nei criteri possiamo utilizzare dati di tipo testuale, numerico o data/ora, come illustrato dalla tabella seguente.

OPERATORE	SIGNIFICATO	ESEMPIO
"criterio"	Si tratta di un testo	"smartphone"
criterio	Si tratta di un numero	35,4
#criterio#	Si tratta di una data	#01/04/2015#

Il criterio, inoltre, è a tutti gli effetti una condizione che usa un operatore di confronto. La seguente tabella mostra il significato dei principali operatori di confronto.

OPERATORE	SIGNIFICATO	ESEMPIO
=valore	Uguale	=2
>valore	Maggiore	>2
>=valore	Maggiore o uguale	>=2
<valore	Minore	<2
<=valore	Minore o uguale	<=2
<>valore	Diverso	<>2
condizione1 and condizione2	Entrambe le condizioni devono essere verificare	>=#01/01/2010# AND <=#31/12/2010#
condizione1 or condizione2	Almeno una condizione deve essere verificata	"Torino" OR "Verona"
Not condizione	La condizione non deve essere verificata	NOT "Milano"
Is Null	Il campo non contiene valori	
Between valore1 and valore2	I valori sono compresi tra valore1 e valore2	between 10 and 100
Like "stringa*"	Solo per dati testuali che vanno racchiusi tra doppi apici. I valori iniziano con la stringa che precede l'asterisco	like "a*"

4. Trasciniamo il campo fornitore nella prima colonna, quindi posizioniamoci nella casella Criteri e inseriamo la seguente condizione di ricerca (criterio) necessaria per ottenere tutti i record di marca "findus":
   ="findus"

5. Adesso trasciniamo anche i campi nome, imponibile, scorta e giacenza nelle colonne successive.
6. Chiudiamo e salviamo la query con il nome marca_findus.

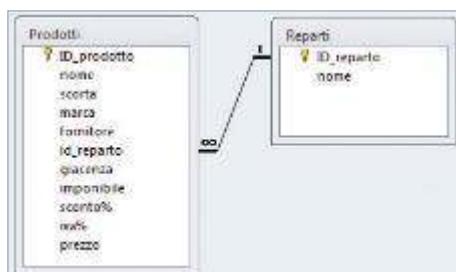
7. Verifichiamo ora il risultato della query: per fare questo, dobbiamo mandare in esecuzione l'interrogazione, facendo clic con il tasto destro sulla query, quindi su Apri.

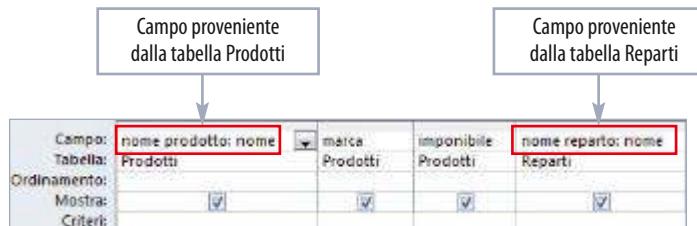
Possiamo aprire l'oggetto Query, analogamente a quanto già visto per le tabelle, mediante il doppio clic sull'icona che rappresenta l'interrogazione.

8. Il risultato è un elenco di cinque colonne (i campi elencati sono quelli trascinati nel punto 5). Come possiamo notare, appare un elenco di prodotti della marca indicata nei criteri.

Query parametriche

marca_findus

marca	nome	imponibile	scorta	giacenz
findus	spinaci	€ 2,20	10	9
findus	bastoncini pesce	€ 12,50	10	15
findus	spinaci	€ 1,80	10	3
findus	filetto platessa	€ 21,00	10	20

Per ricercare i record che hanno un determinato campo vuoto dobbiamo inserire la stringa "Is Null" come criterio di ricerca. In tal modo, se per esempio volessimo ricercare tutti i prodotti che non hanno nessun fornitore, dovremmo inserire il seguente criterio nella QBE:

Campo:	fornitore
Tabella:	Prodotti
Ordinamento:	
Mostra:	☑
Criteri:	Is Null

Query su più tabelle

Le query possono a volte utilizzare campi provenienti da tabelle diverse, ma tuttavia in relazione tra loro. La procedura che segue, che utilizza il database Supermarket.accdb, mostra come ottenere l'elenco dei prodotti con indicato accanto il nome del reparto al quale appartengono.

1. Dopo esserci posizionati sulla scheda Crea e aver fatto clic sul pulsante Struttura Query, dobbiamo collocare nella nostra QBE tutte le tabelle coinvolte da questa richiesta, quindi sia la tabella Prodotti che la tabella Reparti. Per fare questo, facciamo doppio clic in sequenza sulle due tabelle.

2. Possiamo notare che le due tabelle sono state collocate all'interno della finestra della QBE.

3. Trasciniamo adesso i campi interessati della tabella Prodotti, cioè nome, marca, imponibile e quelli della tabella Reparti, cioè nome. Come possiamo notare, vi sono due campi con lo stesso nome, cioè il campo nome, che appare sia nella tabella Reparti che nella tabella Prodotti. Per fare in modo che, eseguendo la query, appaiano due intestazioni diverse, possiamo digitare il nome dell'intestazione personalizzata seguita dai due punti e dal nome del campo.

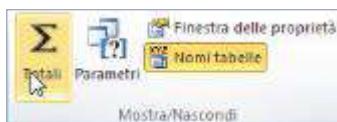
Campo:	nome prodotto: nome	marca	imponibile	nome reparto: nome
Tabella:	Prodotti	Prodotti	Prodotti	Reparti
Ordinamento:				
Mostra:	☑	☑	☑	☑
Criteri:				

4. Eseguendo la query otteniamo l'elenco seguente, dove possiamo notare che i dati provengono dalle colonne "nome prodotto" e "nome reparto".

Query1

nome prodotto	marca	imponibile	nome reparto
merluzzo 125 g	unigel	€ 7,00	surgelati
prezzemolo	pistogel	€ 1,50	surgelati
spinaci	findus	€ 2,20	surgelati

Le query di raggruppamento

Le **query di raggruppamento** sono query di selezione che usano determinate funzioni matematiche per realizzare, per esempio, la somma, la media, il conteggio, il minimo, il massimo ecc. Più precisamente, le query di raggruppamento eseguono due distinte operazioni: la prima è quella di raggruppare i record attraverso un campo, la seconda è quella di eseguire su ciascun gruppo di record l'operazione prescelta tramite una funzione di aggregazione (Somma, Media, Massimo, Minimo, Conteggio ecc.).

Query di raggruppamento
Chiamate anche query di aggregazione, consentono di effettuare calcoli o statistiche sui dati memorizzati nelle tabelle, l'importante è che abbiano qualche caratteristica in comune. Raggruppare significa unire i dati in base a un campo chiamato campo aggregato o campo calcolato. Per esempio, possiamo determinare con questi tipi di query il prezzo medio, la quantità massima o minima, la somma dei prezzi degli ordini ricevuti da un'azienda ecc.

La procedura che segue mostra come realizzare una query di raggruppamento.

1. Prendiamo in esame il database Vendite.accdb, la cui struttura è la seguente.

Come possiamo notare, la tabella Vendite contiene il nome (campo Negozio) dei vari negozi e il relativo importo totale di merci vendute (campo Importo). La tabella Vendite è in relazione con Regioni attraverso la chiave esterna id_regione e con la tabella Categorie attraverso la chiave esterna id_categoria.

2. Vogliamo calcolare la media degli Importi di vendita per ciascuna Regione: per fare questo, effettuiamo un raggruppamento tramite il campo nome_regione, presente nella tabella Regioni, quindi assegniamo al campo Importo la funzione di aggregazione Media.
3. Creiamo una nuova query in modalità struttura, facendo clic sul pulsante Struttura query presente nel gruppo Query della scheda Crea.
4. Collochiamo, nella parte superiore della QBE, le tabelle Regioni, Vendite, Categorie, facendo doppio clic su di esse.

5. Facciamo adesso clic sul pulsante Totali presente nel gruppo Mostra/nascondi della scheda Progettazione per attivare il tipo query di raggruppamento.
6. Trasciniamo il campo nome_regione, della tabella Regioni, nella prima colonna, e selezioniamo nella riga Formula la voce Raggruppamento.
7. Trasciniamo il campo Importo della tabella Vendite, quindi scegliamo Media nella riga Formula che indica la funzione di aggregazione:
8. Dopo aver salvato la query con il nome media_importo_regione, la eseguiamo ottenendo il seguente risultato:

media_importo_regione

nome_regione	MediaDiImporto
Emilia romagna	€ 306.430,00
Friuli	€ 230,00
Lazio	€ 154.558,67
Liguria	€ 345.335,00
Lombardia	€ 187.594,00
Marche	€ 144.772,50
Piemonte	€ 256.235,00
Toscana	€ 8.235,00
Trentino	€ 272.285,33
Veneto	€ 218.805,00

Le query a campi incrociati

Le query a campi incrociati consentono di fornire rapporti di sintesi formati da raggruppamenti organizzati logicamente su due dimensioni, una per le righe e una per le colonne: si tratta di query di raggruppamento che permettono di raggruppare i dati tramite due campi che rappresentano le intestazioni di riga e di colonna.

La procedura che segue mostra come realizzare una query a campi incrociati per ottenere la somma degli importi per regione e categoria.

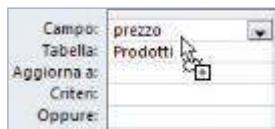
1. Prendiamo in esame il database Vendite.accdb, che abbiamo utilizzato nell'esempio precedente.
2. Creiamo una nuova query in modalità struttura, facendo clic sul pulsante Struttura query presente nel gruppo Query della scheda Crea.
3. Collochiamo, nella parte superiore della QBE, le tabelle Regioni, Vendite, Categorie, facendo doppio clic su di esse.
4. Facciamo clic sul pulsante A campi incrociati presente nel gruppo Tipo di query della scheda Progettazione per attivare il tipo query a campi incrociati.
5. Trasciniamo i campi Denominazione della tabella Categorie e nome_regione dalla tabella Regioni.

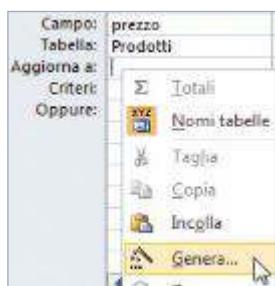
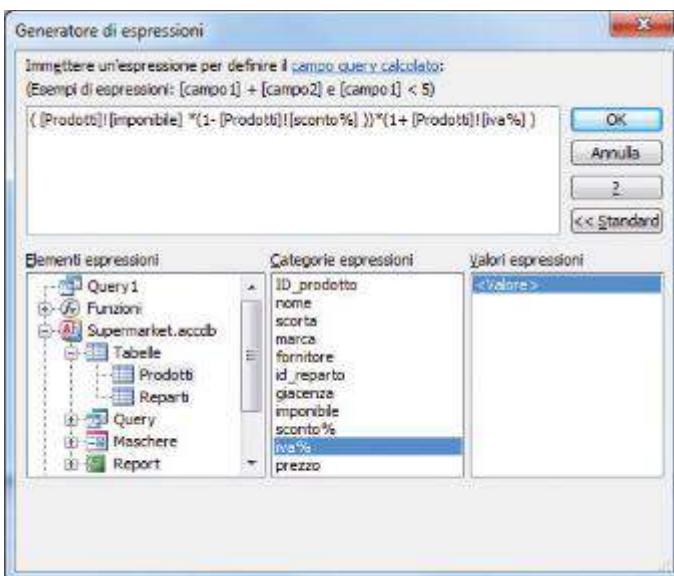
6. Assegniamo al campo Denominazione l'intestazione riga dalla casella Campi incrociati.
7. Assegniamo al campo nome_regione l'intestazione di colonna dalla casella Campi incrociati.

Campo:	Denominazione	nome_regione	Importo
Tabella:	Categorie	Regioni	Vendite
Formula:	Raggruppamento	Raggruppamento	Somma
Campi incrociati:	Intestazione riga	Intestaz. colonna	Valore

8. Assegniamo al campo Importo la Formula di sommatoria (Somma) e il Valore nella casella Campi incrociati.
9. Dopo aver salvato la query con il nome somma_vendite_categorie_regioni, la eseguiamo ottenendo il seguente risultato:

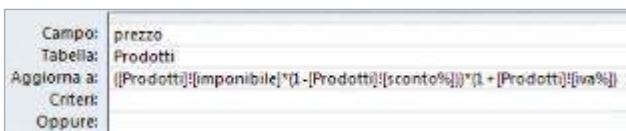
somma_vendite_categorie_regioni

Denominazione	Emilia romagna	Friuli	Lazio	Liguria	Lombardi	Marche	Piemonte	Toscana	Trentino	Veneto
Editoria elettronica	€ 2.893,00									
Hardware	€ 1.833.256,00		€ 5.678,00	€ 690.670,00	€ 2.450,00	€ 568.256,00			€ 816.400,00	
Materiale di consumo							€ 256.235,00			€ 2.010,00
Multimedia			€ 1.768,00		€ 747.926,00	€ 9.156,00				
Software	€ 2.431,00	€ 230,00	€ 456.230,00			€ 1.678,00		€ 8.235,00	€ 456,00	€ 435.600,00

Le query di comando

Quando eseguiamo una query di comando il sistema ci avverte, tramite una finestra di dialogo, che la tabella verrà modificata, informandoci anche del numero di record che saranno influenzati dalla query stessa.
Tra le query di comando esaminiamo le query di aggiornamento, che consentono di modificare automaticamente il contenuto dei record effettuando anche dei calcoli in modo contestuale.
La procedura che segue utilizza il database Supermarket.accdb e mostra come calcolare il prezzo degli articoli partendo dall'imponibile di ogni articolo e collocando il risultato nel campo prezzo.

1. Facciamo clic sul pulsante Struttura query presente nel gruppo Query della scheda Crea per creare una nuova interrogazione.
2. Selezioniamo la tabella Prodotti dalla finestra Mostra tabella.
3. A questo punto selezioniamo il tipo di query che intendiamo creare: facciamo clic sul pulsante Query di aggiornamento presente nel gruppo Tipo di query della scheda Progettazione.

4. Trasciniamo il campo prezzo nella prima colonna della finestra QBE.
   Come possiamo notare, appare una nuova riga denominata Aggiorna a, che conterrà il valore o l'espressione che dovrà essere inserita nei vari record di questo campo.

5. Dobbiamo adesso inserire nella casella Aggiorna a la formula che calcoli il prezzo a partire dall'imponibile, l'aliquota IVA e lo sconto. Per fare questo, facciamo clic con il tasto destro del mouse nella casella Aggiorna a e selezioniamo la voce Genera...
6. Appare la finestra chiamata generatore di espressioni, che consente di inserire agevolmente una formula. Nel nostro caso, la formula che dobbiamo inserire per poter calcolare il prezzo è la seguente:

(imponibile * (1 – sconto%)) * (1 + iva%)

Possiamo digitare la formula interamente a mano, ricordando di inserire i nomi dei campi della tabella tra parentesi quadre. In alternativa possiamo far inserire automaticamente nella formula i nomi dei campi delle tabelle facendo doppio clic sulla tabella interessata (finestra Elementi espressioni) e sul relativo campo (finestra Categorie espressioni).

AREA DIGITALE

Query di cancellazione, creazione tabella e accodamento

7. Facendo clic su OK la formula viene collocata nella casella Aggiorna a.

8. Mandando in esecuzione la query otteniamo l'aggiornamento di tutti i record della tabella al valore calcolato tramite la formula.

METTITI ALLA PROVA

• Query di raggruppamento • Query di aggiornamento

APRI IL FILE Magazzino.accdb

1 Crea una query di comando per calcolare la media della quantità per tipo di movimento (carico e scarico).

2 Crea una query che riduca del 10% il prezzo dei prodotti che hanno giacenza inferiore a 30 e che aumenti del 10% il prezzo dei prodotti con scorta maggiore di 10.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Una query di Access consente di:
- **a** filtrare una maschera sulla base di alcuni criteri
- **b** filtrare una tabella sulla base di alcuni criteri
- **c** effettuare delle stampe di una tabella raggruppate in ordine ad un campo
- **d** modificare la struttura di una relazione

2 Quali criteri tra i seguenti sono applicabili solo a campi numerici? *(2 risposte)*
- **a** like
- **b** <>
- **c** Is Null
- **d** between

3 Quale tra le seguenti non è una query di comando?
- **a** Query di cancellazione
- **b** Query a campi incrociati
- **c** Query di aggiornamento
- **d** Query di creazione tabella

4 In una query di raggruppamento quale fase tra le seguenti viene effettuata per prima?
- **a** Calcolo
- **b** Funzione di aggregazione
- **c** Raggruppamento
- **d** Criteri

5 Quali tra le seguenti non è una funzione di aggregazione?
- **a** Somma
- **b** Dove
- **c** Conteggio
- **d** Maggiore o uguale

6 Come si indica un parametro in una query?
- **a** Tra parentesi quadre
- **b** Tra parentesi tonde
- **c** Tra doppi apici
- **d** Tra apici

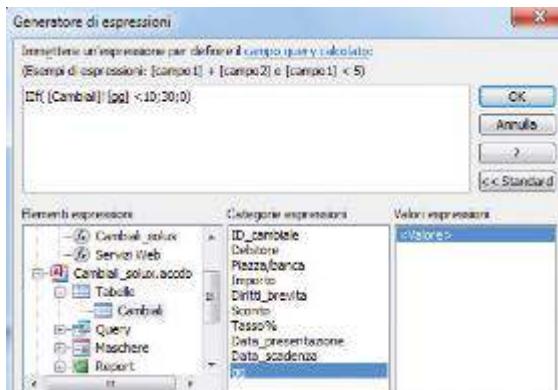
7 Quale tra i seguenti è l'acronimo QBE?
- **a** Query By Exercise
- **b** Query By Example
- **c** Query Base Example
- **d** Query Base Entry

8 Quali tra le seguenti sono query di comando? *(2 risposte)*
- **a** Query di raggruppamento
- **b** Query di aggiornamento
- **c** Query di selezione
- **d** Query di eliminazione

9 Quale tra i criteri seguenti è corretto?
- **a** Like 10 Or 20
- **b** Like 10 And 20
- **c** Between 200 And 100
- **d** Between 100 And 200

10 Quale criterio tra i seguenti indica i record con data di nascita uguale a 10 ottobre 2001?
- **a** ="10/10/2001"
- **b** = #10/10/2001#
- **c** ='10/10/2001'
- **d** =10/10/2001

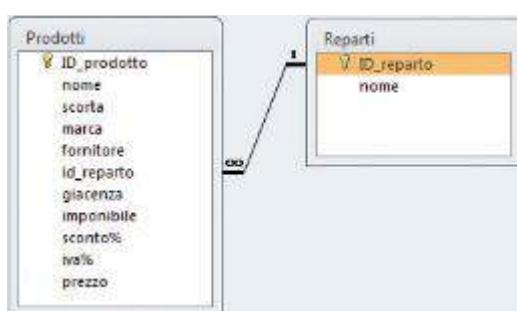
VERIFICA... le competenze

ESERCIZI

1 Apri il file Cambiali.accdb
Il database è formato dalla seguente tabella.

Cambiali: ID_cambiale, Debitore, Piazza/Banca, Importo, Diritti_brevità, Sconto, Tasso%, Data_presentazione, Data_scadenza, gg.

- Crea una query di aggiornamento che calcoli, nel campo gg, i giorni intercorrenti tra le date presenti nei campi Data_presentazione e Data_scadenza.
- Crea una query di aggiornamento che calcoli, nel campo sconto, lo sconto della cambiale utilizzando la formula seguente.

$$\text{sconto} = \frac{\text{tasso * importo * giorni}}{365}$$

- Crea una query di aggiornamento che calcoli l'importo dei diritti di brevità. Se i giorni che intercorrono tra la data di presentazione e la scadenza sono inferiori a 10 viene corrisposto un diritto di 30 €. Per fare questo, inserisci la funzione condizionale IIF(condizione;vero;falso), come indicato nella figura a lato.
- Confronta la tua soluzione con quella presente nel file Cambiali_solux.accdb.

2 Apri il file Supermarket.accdb
Il database è formato dalle seguenti tabelle.

Prodotti: ID_prodotto, nome, scorta, marca, fornitore, id_reparto, giacenza, imponibile, sconto%, iva%, prezzo

Reparti: ID_reparto, nome

Le tabelle sono in relazione come segue.

- Crea una query che mostri tutti i prodotti con imponibile compreso tra 10 e 20 euro.
- Crea una query che mostri i fornitori delle marche "Happiness", "Lagostina" e "Mattel".
- Crea una query che calcoli quanti prodotti ci sono per ciascun reparto, utilizzando le due tabelle (ricorda di collocare entrambe le tabelle, sia Reparti che Prodotti nella parte superiore della QBE).
- Crea una query che mostri tutti i prodotti, con i relativi reparti, che hanno scorta inferiore a 10.
- Crea una query che calcoli la media della giacenza per ciascun reparto.
- Confronta la tua soluzione con quella contenuta nel file supermarket_solux.accdb.

VERIFICA... le competenze

3 Apri il file Movimenti.accdb
Il database è formato dalle seguenti tabelle.

Anagrafica: ID_persona, Cognome, Nome, Provincia, Saldo

Movimenti: ID_movimento, id_persona, Causale, Data, Importo

Causali: ID_causale, Causale

Le tabelle sono in relazione come segue.

- Crea una query che calcoli quante persone hanno effettuato movimenti con causale "BANCOMAT".
- Crea una query che calcoli la media degli importi dei movimenti raggruppati per Provincia.
- Crea una query che elenchi tutti i cognomi e nomi di coloro che hanno effettuato movimenti nel mese di febbraio dell'anno 2017.
- Crea una query che calcoli la media del saldo raggruppato per Provincia.
- Crea una query che conti i movimenti effettuati da persone delle province di Verona e Milano (VR, MI).
- Crea una query che conti le persone della Provincia fornita come parametro.
- Crea una query che calcoli l'importo minimo e massimo dei movimenti di causale "BANCOMAT" e "RICARICA SCHEDA PREPAGATA".
- Crea una query che elenchi il nome e cognome delle persone che hanno effettuato movimenti di almeno 500 euro.
- Crea una query che calcoli la somma degli importi dei movimenti raggruppati per Causale.

4 Crea il file Vendite.accdb
Il database è formato dalle seguenti tabelle.

Regioni: ID_regione, nome_regione

Vendite: id_regione, id_categoria, Negozio, Importo

Categorie: ID_categoria, Denominazione

Le tabelle sono in relazione come segue.

- Crea una query che calcoli l'importo minimo e massimo per negozio.
- Crea una query che calcoli quanti negozi ci sono per regione.
- Crea una query che calcoli la media degli importi per categoria.
- Crea una query che calcoli quanti negozi ci sono per categoria, solo per le categorie "Hardware" e "Multimedia".

VERIFICA... le competenze

5 Crea il file Incassi.accdb
Il database è formato dalle seguenti tabelle.

Regioni: ID_regione, Nome

Comuni: ID_comune, id_regione, Nome

Incassi: ID_incasso, id_comune, Importo, Anno

Le tabelle sono in relazione come segue.

- Crea una query di raggruppamento che mostri la media degli importi per la Regione immessa come parametro.
- Crea una query che elenchi i comuni che hanno realizzato un importo inferiore a 100 nell'anno 2017.
- Crea una query che calcoli l'importo medio degli incassi realizzati per Regione.
- Crea una query a campi incrociati che calcoli la media degli importi per Regioni e Anno, come mostrato dalla figura a lato.

Nome	2014	2015	2016	2017	2018
Ancona					23
Arezzo					45
Avellino					34
Bari				56	98
Bergamo		345	278	298	412
Bologna			57	89	156
Brescia			312	356	478
Como		34	31	124	231
Cuneo				121	232
Fermo					19
Firenze					23
L'Aquila					56
Lecce			76	89	213
Milano		1200	1214	1324	1400
Modena			123	145	178
Napoli		65	90	156	239
Novara					10
Parma				56	78
Pesaro					67
Pescara			55	99	148
Piacenza					12
Reggio Emilia					56
Rimini				56	78
Salerno				109	65
Torino	657	654	678	987	1002
Venezia		29	56	102	145
Verona			26	78	134
Vicenza				121	123

6 Crea il file ContiCorrenti.accdb
- Crea tre tabelle, rispettivamente Clienti, Conti_correnti e Prestiti e mettile in relazione tra di loro come indicato nella seguente figura.

- Popola le tabelle inserendo alcuni record nelle tabelle.
- Crea una query che calcoli il saldo medio dei conti correnti di ciascun cliente.
- Crea una query che visualizzi gli importi in prestito da ciascun cliente.
- Crea una query che calcoli l'importo massimo e minimo di prestiti effettuati dalla banca.
- Crea una query che visualizzi i nomi dei clienti che hanno chiesto prestiti con rateizzazione maggiore di 60 mesi.
- Crea una query che visualizzi i nomi dei clienti che hanno chiesto prestiti con importo tra 20.000 euro e 100.000 euro.

VERIFICA... le competenze

7 Apri il file Pellicole.accdb

– Crea le seguenti tabelle, scegliendo il tipo più adatto a contenere i dati indicati.

Compensi: ID_compenso, id_film, id_attore, Compenso

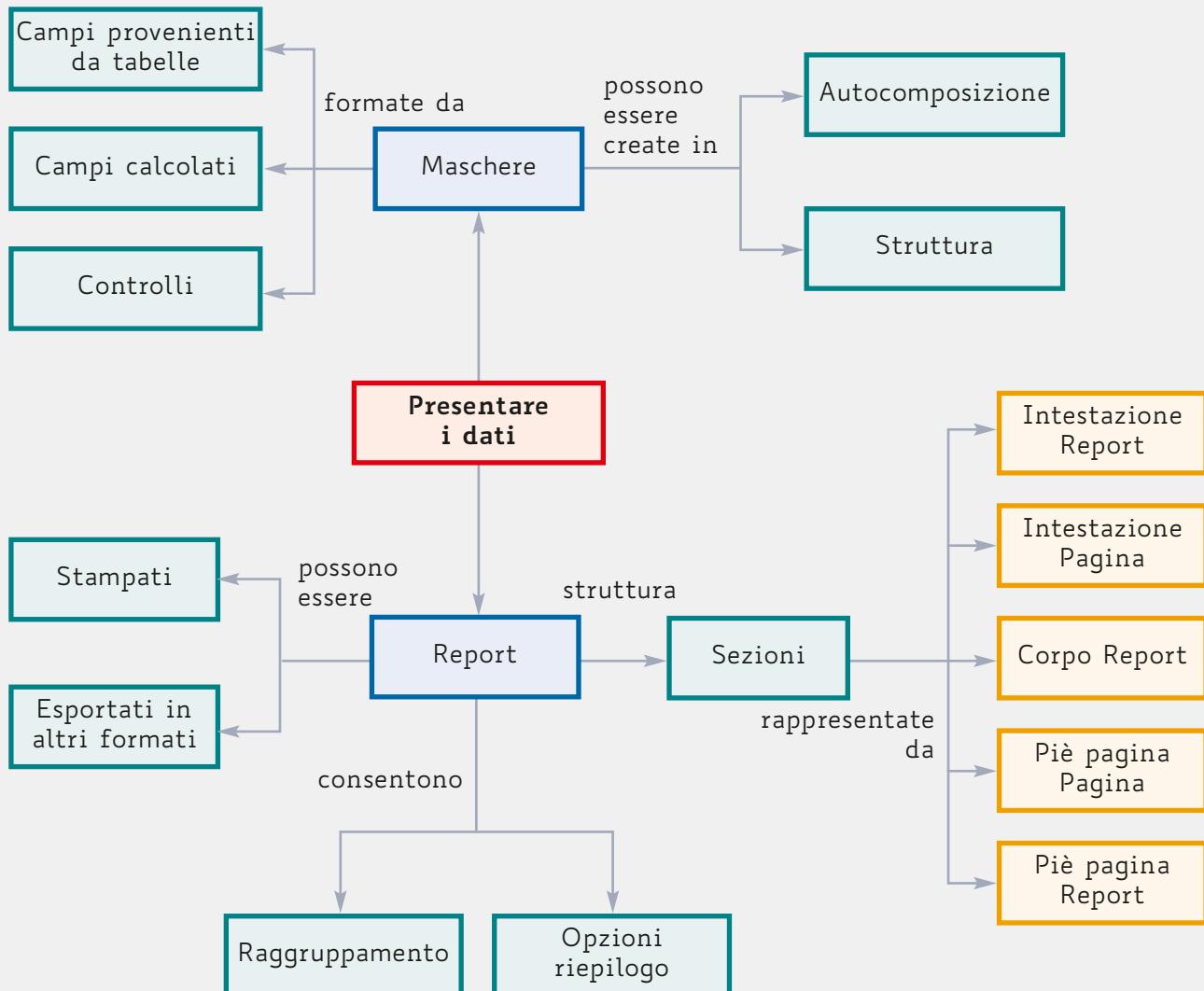
Film: ID_film, Titolo, Regista, Genere, Valutazione

Attori: ID_attore, Cognome, Nome

– Assegna le seguenti relazioni, assegnando i vincoli di integrità che ritieni opportuni.

– Inserisci i seguenti record nelle tabelle Film, Attori e Compensi.

Film				
ID_film	**Titolo**	**Regista**	**Genere**	**Valutazione**
1	La bella e la bestia	Bill Condon	Fantastico	6
2	X-Men - Giorni di un futuro passato	Bryan Singer	Fantastico	4
3	The master	Paul Thomas Anderson	Drammatico	7
4	Noi siamo infinito	Stephen Chbosky	Drammatico	8
5	Aus dem nichts	Fatih Akin	Azione	7
6	Bastardi senza gloria	Quentin Tarantino	Azione	4
7	Troy	Wolfgang Petersen	Storico	6

Attori		
ID_Attore	**Cognome**	**Nome**
1	McKellen	Ian
2	Phoenix	Joaquin
3	Kruger	Diane
4	Watson	Emma
5	Adams	Amy
6	Pitt	Brad

Compensi			
ID_compenso	**id_attore**	**id_film**	**Compenso**
1	1	1	1200
2	4	1	2500
3	1	2	600
4	5	3	1000
5	2	3	1500
6	4	4	670
7	6	6	3400
8	3	6	450
9	3	5	230
10	3	7	450
11	6	7	1300

Le maschere e i report

IN QUESTA LEZIONE IMPAREREMO...

- a creare maschere personalizzate
- a realizzare maschere complesse con sottomaschere
- a realizzare prospetti (report)

MAPPA CONCETTUALE

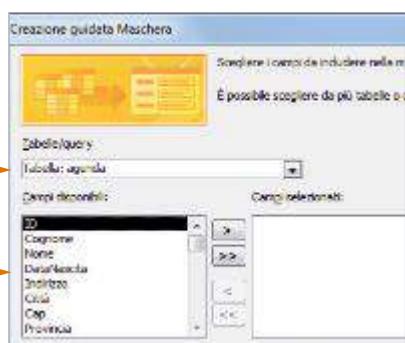
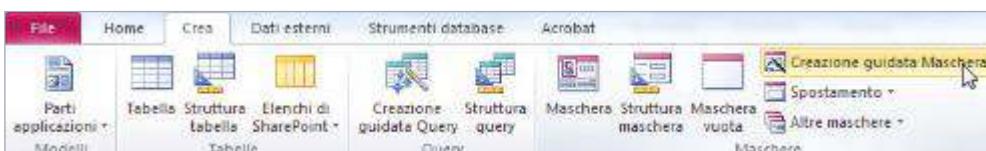
Le maschere

Maschere
La maschera è un oggetto di Access che consente di mostrare, modificare e immettere i record attraverso una forma esteticamente più gradevole rispetto alla Visualizzazione Foglio dati. Inoltre, consente l'immissione di controlli speciali (pulsanti, caselle combinate, caselle di testo ecc.), che possono essere comandati da procedure chiamate Macro.

Le **maschere** sono particolari finestre che permettono di mostrare moduli che possono essere compilati per inserire dati nelle tabelle.
Una maschera è un'interfaccia grafica che agevola la gestione delle tabelle, aggiungendo, modificando o eliminando i dati contenuti nelle tabelle stesse.
È formata principalmente da campi, composti da due elementi:

- casella di testo;
- etichetta.

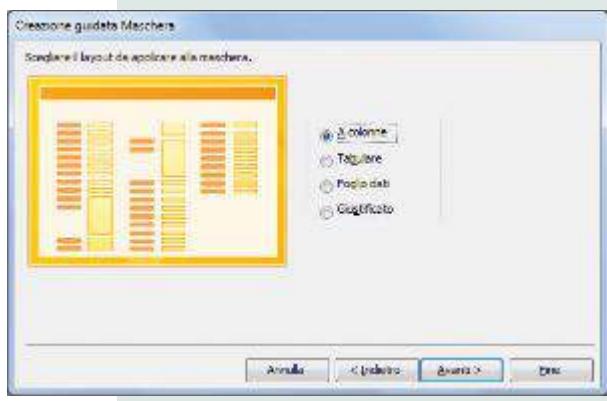
La figura che segue mostra una maschera in Visualizzazione Struttura.

Se apriamo la maschera, appare la finestra mostrata a lato, dove l'utente può immettere o modificare i dati della tabella, presentati all'interno di caselle di testo.

Le maschere sono utili anche ai fini della riservatezza dei dati: aprendo infatti una tabella tramite Foglio dati, l'utente può accedere a tutti i record memorizzati nella tabella. Attraverso l'uso di una maschera, invece, è possibile escludere alcuni campi o record dalla vista.

Creare una maschera in autocomposizione

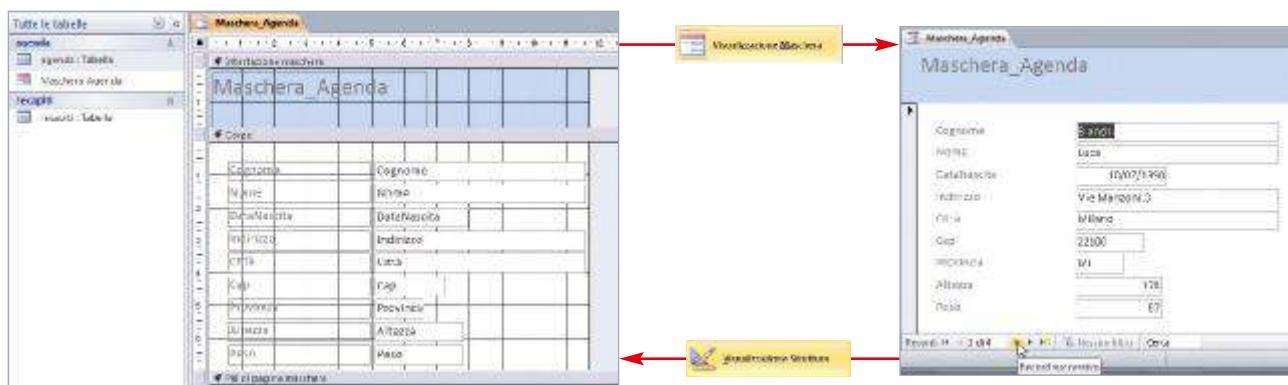
Le maschere possono essere create sia in autocomposizione che manualmente.
La procedura che segue illustra come creare una maschera con Access in autocomposizione per l'inserimento e la modifica dei record del database Agenda.accdb.

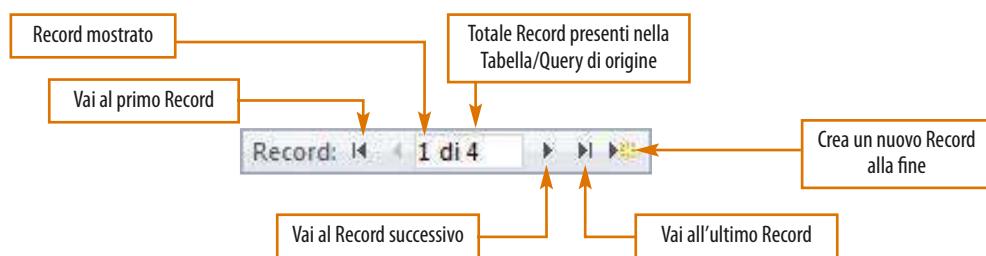
1. Una Maschera è un oggetto di Access, esattamente come una Tabella o una Query. Per attivarne l'autocomposizione dobbiamo posizionarci nella scheda Crea e selezionare il pulsante Creazione guidata Maschera dal gruppo Maschere.

2. A questo punto possiamo selezionare la tabella o la query origine della Maschera. In questo caso, utilizziamo come **origine dati** la tabella Agenda.

Origine dati
Con l'espressione "origine dati" vengono indicati tutti i dati che provengono dalla tabella o dalla query prescelta.

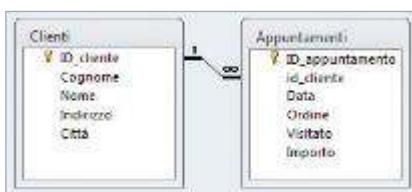
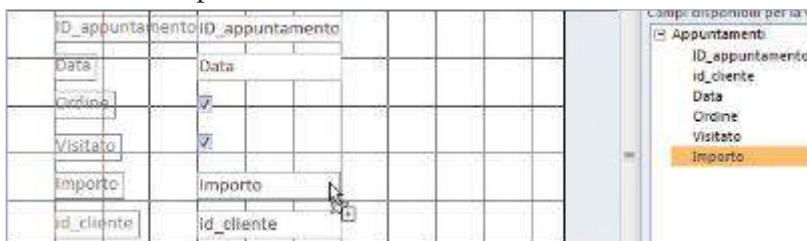
3. Scegliamo adesso quali campi mostrare nella maschera selezionandoli dall'elenco della sezione Campi disponibili, quindi facciamo clic sul pulsante di aggiunta (>>). In questo caso, non inseriamo il campo chiave primaria rappresentato dal campo di tipo Numerazione automatica chiamato ID. Al termine facciamo clic su Avanti.

4. Nella finestra seguente selezioniamo il layout della maschera, in questo caso di tipo A colonne. Facciamo quindi clic su Avanti per proseguire.

PER SAPERNE DI PIÙ

I LAYOUT DELLA MASCHERA

Per il layout della maschera è possibile scegliere tra diverse tipologie:

– A colonne riporta tutti i campi in colonna, con accanto la descrizione.
– Tabulare mostra i campi in colonna.
– Foglio dati crea una finestra identica a quella del foglio dati (che può essere modificata).
– Giustificato mostra i campi in una sequenza a partire da sinistra verso destra (continuando sulla riga successiva).

5. Nella finestra successiva viene richiesto il nome da assegnare al nuovo oggetto di tipo Maschera: in questo caso, la chiameremo Maschera_Agenda. Il campo sottostante richiede se aprire direttamente la maschera creata oppure se mostrarla in modalità Struttura: in questo caso, scegliamo la seconda ipotesi e facciamo clic su Fine per concludere l'autocomposizione.

6. Otteniamo la schermata che mostra la struttura interna della maschera. Per personalizzare la maschera possiamo passare da Visualizzazione Maschera a Visualizzazione Struttura come mostrato nell'immagine, usando i pulsanti Visualizzazione Struttura e Visualizzazione Maschera presenti nella scheda Home.

Le Maschere consentono di visualizzare i record in esse presenti scorrendoli tramite una apposita barra per lo spostamento dei record.

Creare una maschera personalizzata

Nel caso volessimo progettare una maschera senza utilizzare l'autocomposizione, dobbiamo aprirla in modalità Struttura.

La procedura che segue illustra come creare una maschera personalizzata collocandovi direttamente i campi del database Appuntamenti.accdb, che ha la seguente struttura.

1. Per prima cosa, attiviamo la creazione personalizzata di una maschera attraverso il pulsante Struttura maschera presente nella scheda Crea.

2. Appare una griglia in cui è possibile collocare i campi delle tabelle del database, oltre ad altri oggetti chiamati Controlli. Per mostrare l'elenco delle tabelle e dei vari campi che possono essere trascinati nella maschera, facciamo clic sul pulsante Aggiungi campi esistenti del gruppo Strumenti.

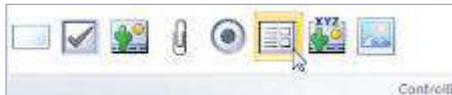
3. Nella parte destra dello schermo appare un riquadro che mostra l'elenco delle tabelle del database e dei relativi campi: in questo caso, sono presenti sia la tabella Appuntamenti che la tabella Clienti.

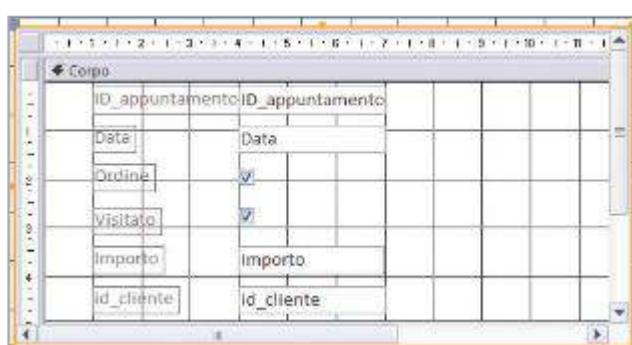
4. Collochiamo adesso il campo prescelto all'interno della maschera: per fare questo, è sufficiente trascinare il campo all'interno della maschera, rilasciandolo nel punto desiderato. Trasciniamo quindi tutti i campi della tabella Appuntamenti fino a ottenere il risultato seguente, cercando di allinearli il più possibile per conferire alla maschera un aspetto uniforme.

5. Come possiamo notare, nella griglia della maschera, per ciascun campo, vengono collocate sia l'etichetta che la casella di testo. Nella figura riportata a lato, possiamo vedere come cambia l'aspetto sia dell'etichetta (in questo caso selezionata) che della casella di testo quando passiamo dalla Visualizzazione Struttura alla Visualizzazione Maschera.

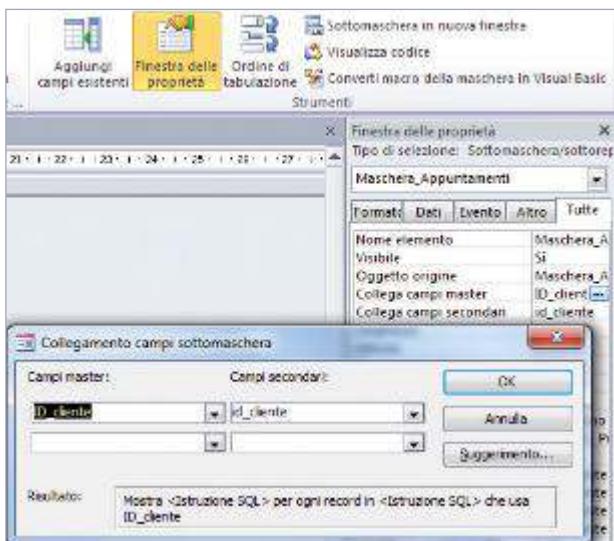
6. Salviamo la maschera con il nome Maschera_Appuntamenti.

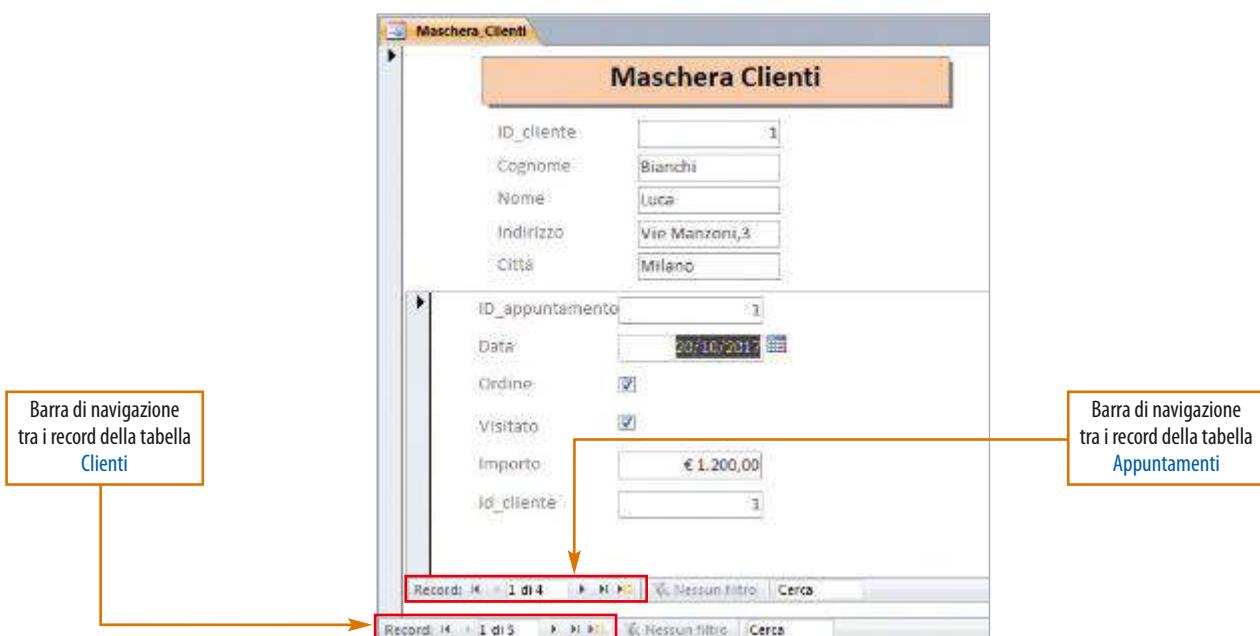
Se utilizziamo la creazione mediante il pulsante Maschera Vuota, presente nel gruppo Maschere della scheda Crea, i Controlli vengono allineati automaticamente sia orizzontalmente che verticalmente. In questo caso, il layout è stato pensato come una tabella dove ogni cella contiene un controllo. In tal modo possiamo spostare tutta la tabella utilizzando il cursore di spostamento blocco posto in alto a sinistra.

Creare una sottomaschera

Illustriamo adesso, nella procedura che segue, come creare una seconda maschera per la tabella Clienti.

1. Per procedere alla creazione della nuova maschera utilizziamo ancora una volta la Visualizzazione Struttura. Prima di tutto, vediamo come inserire un titolo mediante un'etichetta personalizzata, utilizzando i Controlli presenti nella scheda Progettazione.

2. Aggiungiamo adesso un'etichetta dal titolo "Maschera Clienti": per fare questo, trasciniamo il controllo Etichetta nella maschera, e digitiamo al suo interno il titolo.
3. Aggiungiamo un colore di sfondo e una ombreggiatura all'etichetta. Per modificare le impostazioni del controllo, facciamo clic con il tasto destro su di esso, quindi selezioniamo la voce che ci interessa, in questo caso Colore riempimento/sfondo e Aspetto.

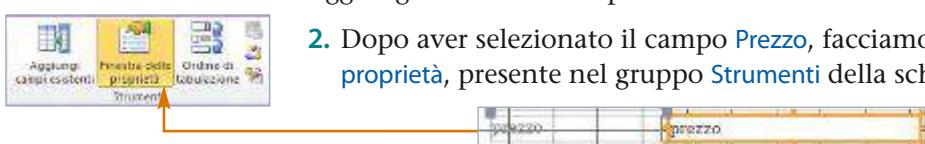
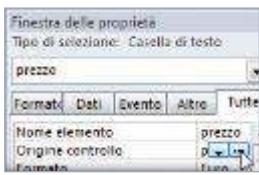
4. A questo punto, aggiungiamo una Sottomaschera per poter visualizzare nella maschera principale i dati di un cliente e contemporaneamente gli appuntamenti di quel cliente nella **sottomaschera**.

Sottomaschera
Lo strumento Sottomaschera consente di visualizzare due maschere distinte all'interno della stessa maschera. In tal modo, possiamo visualizzare i dati relativi a due distinte tabelle. Il contenuto delle sottomaschere può tuttavia venire collegato al contenuto di un campo/casella di testo della maschera principale.

5. Per creare una sottomaschera dobbiamo selezionare la voce Sottomaschera/sottoreport presente nel gruppo Controlli della scheda Progettazione.

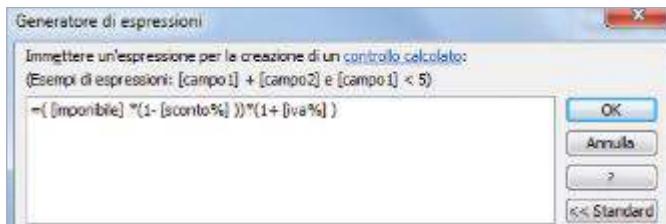
6. A questo punto, trasciniamo il cursore per creare un'area adeguata, necessaria per contenere la maschera Maschera_Appuntamenti creata in precedenza.
7. Appare una finestra nella quale dobbiamo selezionare l'origine: facciamo clic sulla maschera Maschera_Appuntamenti.
8. Come possiamo notare, Maschera_Appuntamenti viene collocata all'interno della sottomaschera appena creata.

9. Adesso, per fare in modo che l'utente possa scorrere i record della sottomaschera, per ciascun record della maschera principale dobbiamo associare le tabelle coinvolte nella sottomaschera

mediante un campo comune, chiamato campo master. Facciamo clic sul pulsante Finestra delle proprietà, presente nel gruppo Strumenti della scheda Progettazione: sul pulsante posto accanto alla casella Collega campi master appare la una finestra dove vengono indicati i campi oggetto di relazione tra le tabelle Clienti (ID_cliente) e Appuntamenti (id_cliente).

> Access riconosce automaticamente i campi master (ID_cliente) in quanto presenti nella tabella che possiede il campo chiave primaria, e i campi secondari in quanto presenti nella tabella che possiede il campo chiave esterna (id_cliente).

10. Eseguendo la maschera appena salvata (Maschera_ Clienti) appare la schermata riportata sotto. La finestra di esecuzione della maschera mostra all'interno della sottomaschera solo gli appuntamenti relativi al cliente sul quale ci siamo posizionati.

I campi calcolati

Un campo calcolato è una casella di testo che contiene una formula. Per creare un campo calcolato dobbiamo quindi collocare nella maschera una nuova casella di testo alla quale associare una formula.

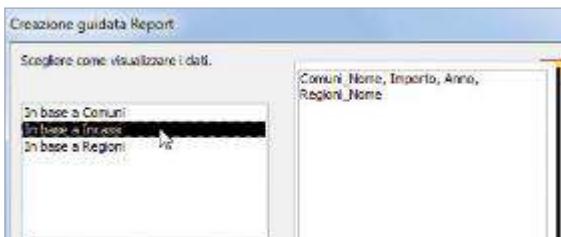
La procedura che segue mostra come inserire una formula per il calcolo del prezzo a partire dal database Supermarket.accdb.

1. Creiamo una nuova maschera tramite il pulsante Creazione guidata Maschera, presente nella scheda Crea. Aggiungiamo tutti i campi della tabella Prodotti in Visualizzazione A colonne.

2. Dopo aver selezionato il campo Prezzo, facciamo clic sul pulsante Finestra delle proprietà, presente nel gruppo Strumenti della scheda Progettazione.

3. Facciamo clic sul pulsante con tre puntini posto accanto alla voce Origine controllo.
4. A questo punto, possiamo inserire la formula di calcolo del prezzo all'interno della finestra delle espressioni, ricordando di scrivere la formula iniziando dal simbolo uguale.

5. Una volta scritta la formula, essa verrà visualizzata sia nella proprietà Origine controllo, che all'interno della casella di testo. Inoltre, i calcoli inclusi in maschere o report verranno eseguiti in tempo reale quando i dati da cui dipendono saranno aggiornati.
6. Quando apriamo la maschera possiamo notare che, a mano a mano che immettiamo i dati, appare il risultato della formula nel campo prezzo.

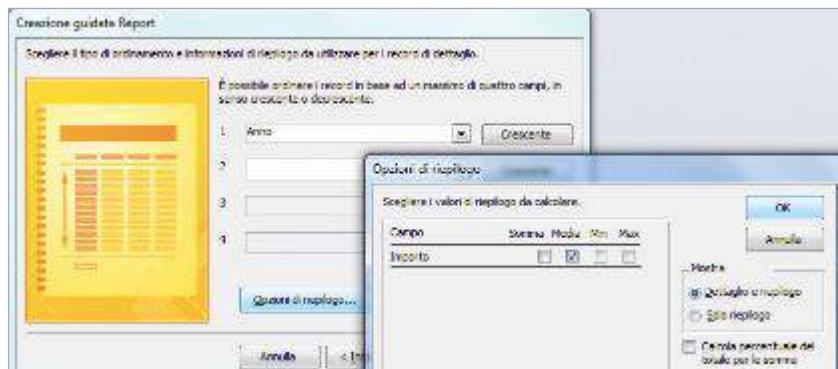
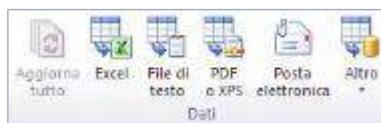
I report

I report (prospetti) consentono di effettuare un riepilogo dei dati presenti in una tabella oppure in una query. Un report consente di organizzare e formattare informazioni, utilizzando elementi sia grafici (immagini, linee, caselle, tabelle, grafici) sia testuali. Il tipo di elementi e la relativa disposizione dipendono dalle impostazioni dell'utente. Alcune applicazioni tipiche dei report sono le etichette postali, le fatture commerciali, i riepiloghi di vendita, i listini ecc.

I prospetti consentono anche di effettuare statistiche in base a raggruppamenti secondo una tecnica chiamata a **rottura di codice**.

Il raggruppamento produce una serie di gruppi di record su ciascuno dei quali è possibile applicare delle statistiche, quali, per esempio, un conteggio oppure la media di un prezzo. Rispetto alle query e alle tabelle, i report consentono una migliore visualizzazione delle informazioni dal momento che visualizzano i record in modo più gradevole.

Rottura di codice

La rottura di codice è una tecnica di ricerca per mezzo della quale si ottiene un elenco con i record raggruppati in un campo chiamato gruppo. Per fare questo, la tabella viene innanzitutto ordinata tramite il campo gruppo, quindi vengono presi in esame i record fino alla "rottura", cioè al cambiamento del contenuto del campo. A ogni rottura del campo si viene a creare un nuovo gruppo.

La procedura che segue illustra la realizzazione di un report nel quale vengano mostrati gli incassi relativi ad alcuni negozi suddivisi per regione di appartenenza, utilizzando il database Incassi.accdb, la cui struttura è la seguente:

1. Facciamo clic sul pulsante Creazione guidata Report presente nel gruppo Report della scheda Crea per avviare l'autocomposizione del nostro prospetto personalizzato.
2. Selezioniamo i campi Nome dalla tabella Regioni, Nome dalla tabella Comuni, Importo e Anno dalla tabella Incassi, quindi facciamo clic su Avanti
3. Durante questa procedura il sistema esamina le tabelle selezionate e individua le relazioni raggruppando in base a esse i record del report. Nella finestra che appare, dobbiamo scegliere in

base a quale tabella raggruppare i record, in questo caso scegliamo la tabella Incassi per poter selezionare manualmente i livelli di gruppo nella finestra successiva. Facciamo quindi clic su Avanti per proseguire.

4. Nella finestra che appare è possibile selezionare il campo mediante il quale effettuare il raggruppamento: in questo caso scegliamo il campo Regioni_Nome. Selezioniamo quindi un secondo livello di gruppo facendo clic sul campo Comuni_Nome, ottenendo il risultato mostrato nella figura a lato.
5. La finestra successiva consente di ordinare l'elenco che verrà visualizzato all'interno di ciascun gruppo. In questo caso scegliamo l'Anno. Attraverso il pulsante Opzioni di riepilogo possiamo scegliere le informazioni di carattere statistico da mostrare accanto alla sezione di riepilogo del gruppo. In questo caso scegliamo la Media dell'Importo.

6. Dopo aver confermato facendo clic su OK, quindi su Avanti, possiamo scegliere la formattazione con cui mostrare i dati del report, come per esempio il layout e l'orientamento.
7. Dopo aver fatto clic su Avanti, si apre la finestra che termina l'autocomposizione: dobbiamo selezionare il nome del report e decidere se visualizzarlo in anteprima, oppure modificarne la struttura. In questo caso, decidiamo di mostrarne l'anteprima.
8. A questo punto, il report può essere stampato oppure esportato in uno dei formati presenti nel gruppo Dati della scheda Anteprima di stampa.

PER SAPERNE DI PIÙ

LA STRUTTURA DI UN REPORT

Analogamente a quanto visto per le tabelle e le query, un report può essere aperto in modalità Struttura per poterlo modificare. Il prospetto appare suddiviso in cinque sezioni:

1 Intestazione Report, che viene visualizzata o stampata una sola volta all'inizio del documento.
2 Intestazione Pagina, che viene visualizzata o stampata per ogni pagina del Report.
3 Corpo, che viene visualizzata ripetuta per ogni record proveniente dalla tabella o query; questa sezione può essere a sua volta suddivisa in altre sezioni (intestazione e piè pagina) per ciascun livello di raggruppamento presente.
4 Piè di pagina pagina, che viene visualizzata o stampata nel fondo di ogni pagina.
5 Piè di pagina report, che viene visualizzata o stampata soltanto una volta e alla fine del documento.

Queste sezioni sono sempre presenti in ogni report: naturalmente è possibile abilitarle o disabilitarle a seconda delle necessità. Inoltre, possono essere presenti sezioni "aggiuntive" per ogni livello di raggruppamento (di solito un "campo", oppure una combinazione di uno o più campi). In ciascuna sezione è possibile inserire controlli di ogni genere, anche se i più comuni sono le etichette e le caselle di testo.

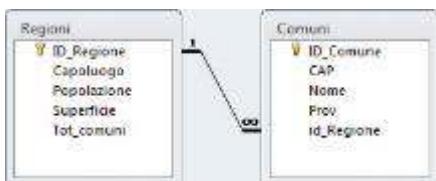
METTITI ALLA PROVA

➜ • Creare maschere personalizzate • Applicare le sottomaschere

APRI IL FILE Supermarket.accdb

1 Crea una maschera principale che mostri i campi della tabella Reparti.
2 Modifica la maschera creata al punto **1** aggiungendo una sottomaschera che mostri tutti i prodotti del reparto selezionato nella maschera principale.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 **Un report può essere generato da:** *(2 risposte)*
- **a** una maschera
- **b** una query
- **c** una tabella
- **d** una macro

2 **Quale tra i seguenti non è un controllo utilizzabile nelle maschere?**
- **a** Pulsante
- **b** Etichetta
- **c** Casella di testo
- **d** Gruppo

3 **Quali tra le seguenti operazioni non possono essere effettuate da una maschera?** *(2 risposte)*
- **a** Modificare la struttura di una query
- **b** Inserire record in una tabella
- **c** Modificare i dati presenti in una tabella
- **d** Modificare la struttura di una tabella

4 **Quale tra le seguenti è una origine dati di una maschera?**
- **a** Una query
- **b** Una maschera
- **c** Una macro
- **d** Un report

5 **Una maschera garantisce la riservatezza dei dati in quanto:**
- **a** consente di proteggere i dati della tabella con password
- **b** permette di mostrare i dati criptati
- **c** consente di escludere alcuni campi o record dalla vista dell'utente
- **d** permette di mostrare i dati solo ad alcuni utenti

6 **Quale tra i seguenti layout di maschera mostra i dati in sequenza a partire da sinistra verso destra, riempiendo tutti gli spazi?**
- **a** Giustificato
- **b** Tabulare
- **c** A colonne
- **d** Foglio dati

7 **Quale affermazione tra le seguenti riguardo le sottomaschere è corretta?** (*2 risposte*)
- **a** I campi secondari vengono riconosciuti automaticamente in quanto sono chiavi primarie
- **b** I campi master vengono riconosciuti automaticamente in quanto sono chiavi primarie
- **c** I campi secondari vengono riconosciuti automaticamente in quanto sono chiavi esterne
- **d** I campi master vengono riconosciuti automaticamente in quanto sono chiavi esterne

8 **Quale tra le seguenti operazioni non è ammessa tra le opzioni di riepilogo dei report?**
- **a** Somma
- **b** Media
- **c** Conteggio
- **d** Min e Max

9 **Una maschera può essere generata a partire da:**
- **a** un report
- **b** un'altra maschera
- **c** una macro
- **d** una query

10 **Quale tra le seguenti azioni non può essere effettuata aprendo un report in modalità struttura?**
- **a** Modificare il corpo
- **b** Aggiungere un piè di pagina
- **c** Modificare l'intestazione pagina
- **d** Cancellare un record

VERIFICA... le competenze

ESERCIZI

1 Apri il file Comuni.accdb

Il database, che contiene i dati relativi a tutti Comuni e le Regioni italiane, presenta la struttura relazionale riportata a lato.

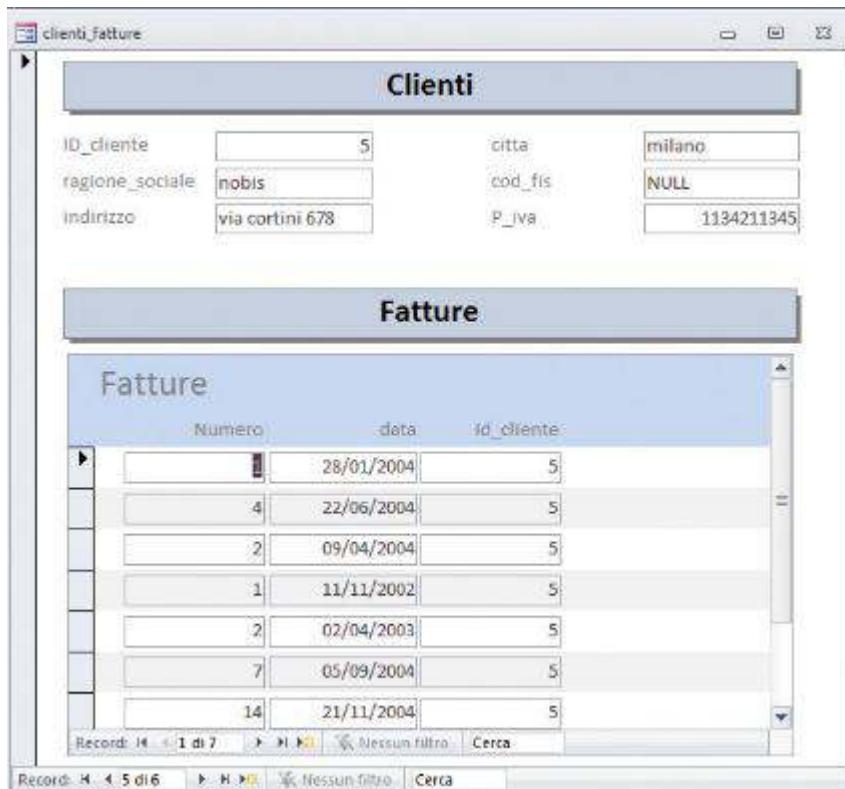
– Crea un report che mostri per ciascuna Regione l'elenco dei relativi Comuni in ordine crescente in base alla Provincia.
– Crea un report che raggruppi per ID_Regione, quindi per Provincia, indicando i dati dei Comuni.
– Crea una maschera che mostri due maschere: in quella principale devono apparire i dati della Regione, mentre nella sottomaschera quelli dei Comuni secondo un layout A colonne.
– Crea un report che raggruppi per Capoluogo, elencando tutti i dati dei Comuni in ordine di CAP, per le sole Regioni con almeno 200 Comuni (Per fare questo il report deve avere come origine dati la query di selezione).

2 Apri il file Vendite.accdb

– Crea un report che mostri le vendite effettuate per ciascuna Regione calcolando il riepilogo dell'importo medio e massimo.
– Crea un report che mostri le vendite effettuate per ciascun reparto calcolando il riepilogo dell'importo minimo.
– Crea un report che mostri le vendite effettuate per Regione, in ordine di negozio, per le sole Regioni che iniziano per "L".

3 Apri il file Magazzino.accdb

Il database ha la seguente struttura relazionale.

– Crea un report che mostri la quantità media di carico e scarico per ciascun magazzino.
– Crea un report che raggruppi per prodotti e per tipo di movimento, quindi ordini i record del report per data dei movimenti.
– Crea una maschera che mostri i dati dei magazzini e i relativi prodotti all'interno di una sottomaschera con layout A colonne.

4 Apri il file Incassi.accdb

– Crea una maschera che mostri i dati della tabella Comuni e della tabella Incassi in una sottomaschera.
– Crea un report che raggruppi per anno e mostri il nome del Comune e l'importo includendo come riepilogo la somma degli importi.
– Crea un report che raggruppi per Regione e quindi per Comune, elencando i dati degli incassi, per le sole Regioni Lombardia, Veneto e Campania. Inserisci la media degli importi degli incassi come opzioni di riepilogo (per fare questo il report deve avere come origine la query di selezione).
– Crea una maschera che mostri tutti i dati degli incassi, calcolando in un campo aggiuntivo, di nome "Target", l'importo maggiorato del 20% per l'anno 2018 e per i Comuni che hanno realizzato un incasso superiore a 350 euro (Per fare questo la maschera deve avere come origine la query di selezione).

VERIFICA... le competenze

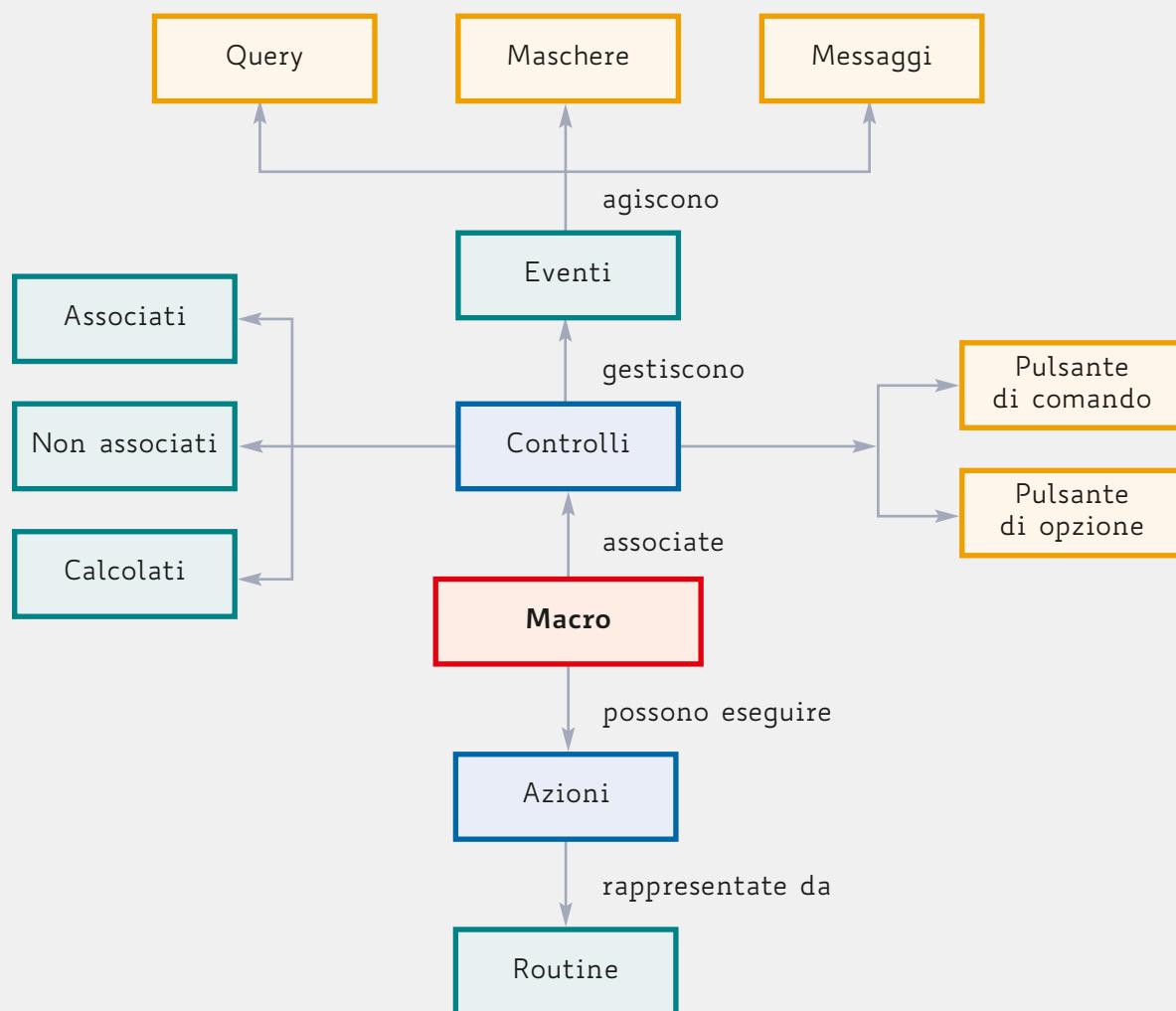
5 Apri il file Fatture.accdb

– Crea una maschera collocandovi tutti i campi della tabella Clienti e una sottomaschera per la tabella Fatture, con layout Tabulare, formattandola come indicato di seguito.

– Confronta la tua soluzione con quella presente nel file Fatture_solux.accdb.

6 Crea il file Esempio_fatture.accdb

– Crea le seguenti tabelle, scegliendo il tipo più adatto a contenere i dati indicati:

Registro_fatture: ID_fattura, Numero, Data, Nome_cliente, Indirizzo_cliente, Imponibile, IVA%, Importo

– Crea una maschera in autocomposizione per mostrare tutti i campi della tabella Registro_fatture, con layout Giustificato.
– Modifica la struttura della maschera creata nel punto precedente aggiungendo una formula che calcoli il totale della fattura all'interno del campo Importo.
– Inserisci alcuni record a piacere utilizzando la maschera creata nel punto 2.

Le macro

IN QUESTA LEZIONE IMPAREREMO...

- a riconoscere e utilizzare i controlli associati, non associati e calcolati nelle maschere
- a creare semplici macro associandole a pulsanti di azione

MAPPA CONCETTUALE

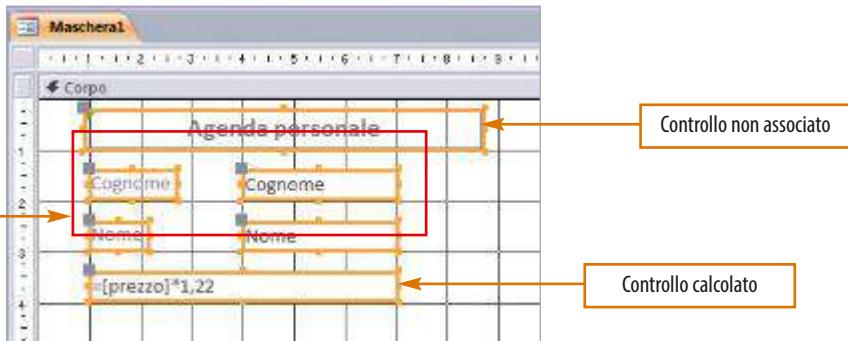
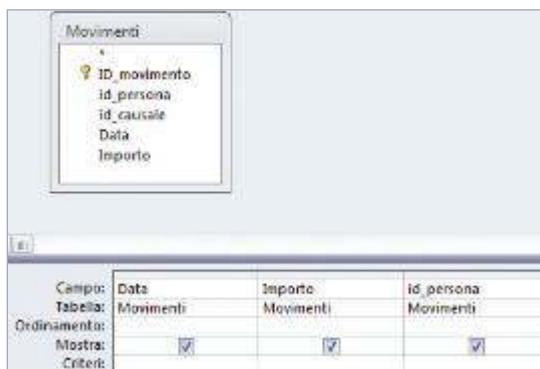
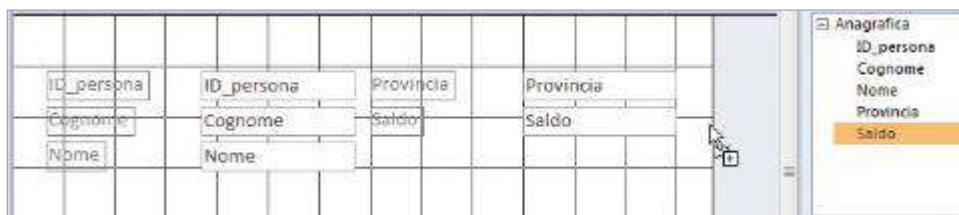
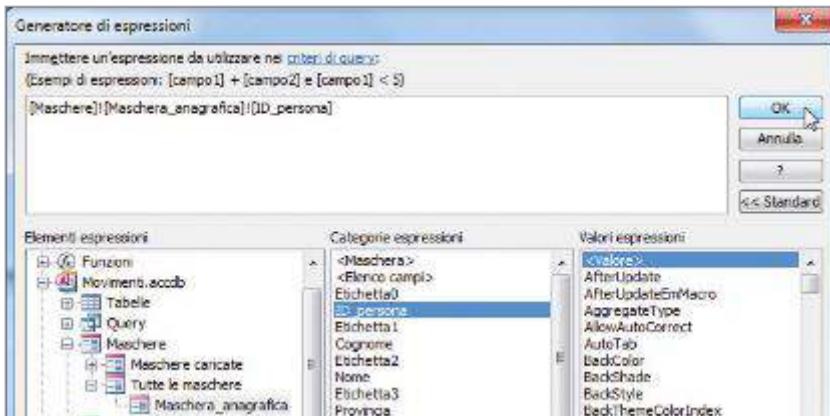
Query
Maschere
Messaggi
agiscono
Eventi
Associati
gestiscono
Pulsante di comando
Non associati
Controlli
Calcolati
associate
Pulsante di opzione
Macro
possono eseguire
Azioni
rappresentate da
Routine

I controlli nelle maschere

Controlli
I controlli sono oggetti che arricchiscono l'interfaccia utente, consentendo anche di eseguire routine, cioè spezzoni di programmi.

Le maschere, chiamate form nell'ambito della programmazione, possono essere progettate, come già accennato nella precedente lezione, in Visualizzazione Struttura: sono formate dalle sezioni Intestazione, Corpo e Piè di pagina e contengono i **controlli**.

I controlli possono essere suddivisi in tre tipologie: associati, non associati, calcolati.

- Controlli associati: sono controlli la cui origine dati è rappresentata da un campo di una Tabella o di una Query. I controlli associati vengono utilizzati per visualizzare valori che provengono direttamente da campi delle tabelle del database, pertanto possono rappresentare dati dei tipi conosciuti (Testo, Numerico, Sì/No, ecc.).
- Controlli non associati: sono controlli che non dispongono di un'origine dati. I controlli non associati vengono utilizzati per visualizzare informazioni, forme e immagini. L'etichetta del titolo di una maschera è un tipico esempio di controllo non associato.
- Controlli calcolati: sono controlli la cui origine dati è rappresentata da un'espressione anziché da un campo. Il valore che si desidera utilizzare viene specificato come l'origine dei dati del controllo tramite la definizione di un'espressione. Un'espressione può essere costituita da una combinazione di operatori (come per esempio +, -, *, (), ecc.), nomi di controlli, nomi di campi, valori costanti e funzioni che restituiscono un valore.

Nella figura che segue sono riportati, all'interno della stessa maschera, i tre tipi di controlli appena descritti: un titolo (controllo non associato) rappresentato da una etichetta, due campi di tipo Testo (controlli associati) e un controllo calcolato rappresentato da una casella di testo.

Associare routine a pulsanti

Il pulsante di comando è un controllo che permette di mandare in esecuzione una routine in base al verificarsi o meno di un evento. L'evento che esegue la routine è il "clic" sul pulsante stesso. Un esempio di routine è rappresentato dall'apertura di una finestra, oppure dall'esecuzione di un calcolo per mostrarne il risultato in una casella di testo.

La procedura che segue utilizza un pulsante di comando per avviare una query che visualizza i movimenti effettuati da una persona con riferimento al database Movimenti.accdb.

1. Dopo aver aperto il database, creiamo una maschera utilizzando il pulsante Struttura maschera della scheda Crea.
2. Collochiamo quindi nella maschera tutti i campi della tabella Anagrafica, che contiene l'elenco dei correntisti, come indicato nella finestra che segue.

3. Una volta completato il lavoro, salviamo la maschera col nome Maschera_anagrafica.

4. Adesso creiamo la query che ci consentirà di ottenere l'elenco dei vari movimenti effettuati dai correntisti. Per fare questo, utilizziamo il pulsante Struttura query della scheda Crea e collochiamo i tre campi Data, Importo, id_persona.

5. Dobbiamo ora collegare la nostra query con la maschera creata in precedenza, per fare in modo che la query mostri solo i movimenti del correntista selezionato nella maschera stessa. A tal fine, immettiamo l'espressione necessaria per ricercare tutti i movimenti: attiviamo il Generatore di espressioni nella casella Criteri del campo id_persona facendo clic con il tasto destro del mouse, quindi facciamo clic su Genera. A questo punto, dalla casella Elementi espressioni selezioniamo Maschera_anagrafica, mentre dalla casella Categorie espressioni selezioniamoil campo ID_persona.

6. Dopo aver confermato con OK, verifichiamo che la nostra espressione venga collocata nella casella Criteri del campo id_persona. Disattiviamo inoltre la casella Mostra, per la evitare che il campo venga visualizzato nella query.

7. Salviamo la query col nome Movimenti_persona.

8. A questo punto apriamo Maschera_anagrafica in Visualizzazione Struttura e collochiamo un pulsante di comando, selezionandolo dal gruppo Controlli della scheda Progettazione.

9. Appena rilasciamo il mouse appare la finestra seguente, nella quale dobbiamo scegliere l'azione da compiere al clic sul pulsante: nella casella Categorie selezioniamo la voce Varie, quindi Esegui query nella casella delle Azioni.

10. Dopo aver fatto clic su Avanti si apre una finestra dove viene richiesto il nome della query che deve essere eseguita: indichiamo la query salvata nel punto 7 della procedura, cioè Movimenti_persona, e facciamo clic su Avanti per proseguire.

11. Nella nuova schermata che appare ci viene richiesto se desideriamo che sul pulsante appaia un testo oppure un'immagine. Scegliamo Testo e digitiamo "Mostra movimenti".

12. Come possiamo notare, il pulsante è completato: salviamo quindi nuovamente la maschera per non perdere il nostro lavoro.

13. Aprendo la maschera possiamo verificare che, facendo clic sul pulsante Mostra Movimenti, appare la query che mostra soltanto i movimenti del record selezionato dalla tabella Anagrafica.

Maschera_anagrafica | Movimenti_persona

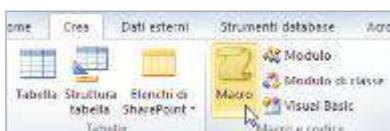
Data	Importo
12/09/2017	€ 340,00
11/01/2018	€ 670,00
08/02/2018	€ 70,00

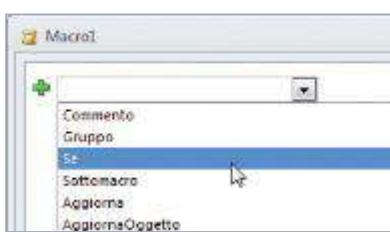
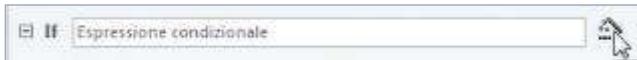
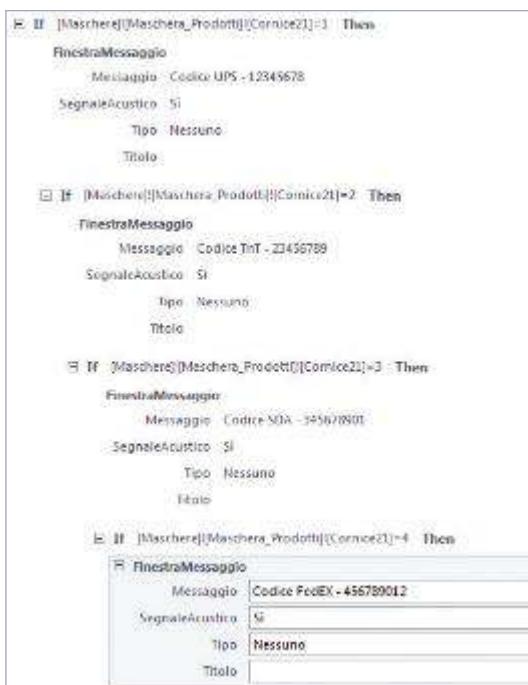
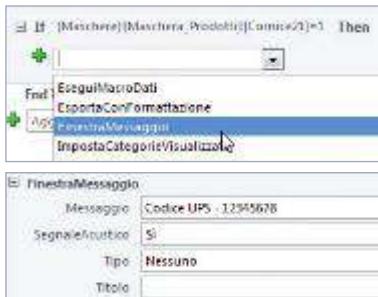
Le macro

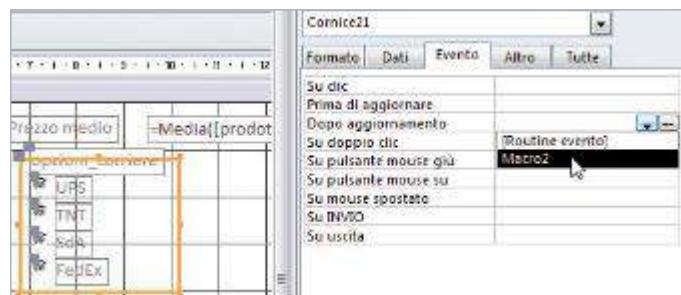
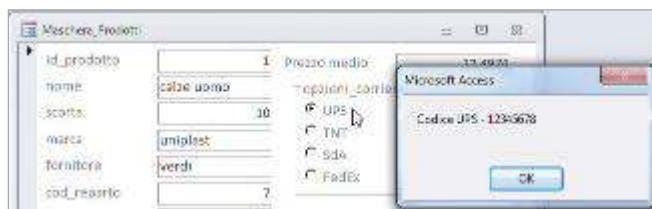
Il controllo a Gruppo di opzioni consente all'utente di effettuare una scelta esclusiva o multipla, mostrando un elenco di voci, raggruppati in una cornice, come mostrato a lato nell'immagine di esempio.

La procedura che segue mostra come realizzare una maschera contenente dei gruppi di opzioni e come associare a essi una **macro**.

Macro

Il termine macro deriva dalla contrazione del termine macroistruzioni, utilizzato nell'ambito della programmazione per indicare delle istruzioni molto potenti formate a loro volta da altre istruzioni. Una macro è una sequenza di istruzioni richiamabile attraverso un controllo oppure una combinazione di tasti. Le macro consentono di eseguire alcune operazioni, chiamate in generale azioni.

1. Con riferimento al database Supermarket.accdb, apriamo Maschera_Prodotti in Visualizzazione Struttura.
2. Adesso dobbiamo collocare un controllo Gruppo di opzioni, presente nel gruppo Controlli della scheda Progettazione.
3. Dopo aver selezionato il controllo, dobbiamo trascinarlo all'interno della maschera, accanto ai campi già presenti; appena viene rilasciato il mouse, appare la finestra a lato, nella quale dobbiamo scrivere l'elenco delle voci di Gruppo di opzioni. In questo caso, vogliamo inserire alcuni nominativi di corrieri di spedizione (UPS, TNT ecc.). Dopo aver scritto l'elenco, facciamo clic su Avanti.

4. Nella finestra successiva viene richiesta la voce predefinita che apparirà a video nel gruppo di opzioni: lasciamo inalterato quanto suggerito dal programma e confermiamo con Avanti.
5. Nella nuova finestra che si apre vengono indicati i valori associati a ciascuna voce selezionata dall'elenco, in questo caso da 1 a 4: facciamo clic su Avanti per continuare.
6. La finestra successiva consente di salvare il valore ottenuto per uso successivo, oppure di memorizzarlo all'interno di un campo presente nella maschera. Scegliamo la prima opzione e facciamo clic sul pulsante Avanti per continuare.

7. Assegniamo infine il nome opzioni_corriere al gruppo di opzioni, che apparirà come etichetta sul bordo della cornice del controllo, quindi facciamo clic sul pulsante Fine per terminare l'autocomposizione.
8. A questo punto, siamo pronti per associare al controllo appena creato la macro che mostrerà il codice identificativo del corriere. Per fare questo, dopo esserci assicurati di aver selezionato il controllo, facciamo clic su Macro, presente nel gruppo Macro e codice della scheda Crea.
9. Come abbiamo già accennato, le macro sono formate da una sequenza di azioni. La macro può essere associata a un controllo mediante un evento, per esempio un clic su di un pulsante oppure, come nel nostro caso, l'aggiornamento del pulsante di opzione (cioè quando selezioniamo una voce dell'elenco).
   Nella finestra che appare dobbiamo selezionare l'azione Se.

10. L'azione Se rappresenta una espressione condizionale che ci consentirà di eseguire una azione in base a una condizione. Facciamo clic sul pulsante di accanto alla casella, rappresentato da una bacchetta magica, per attivare il Generatore di espressioni.

11. All'interno della finestra del Generatore di espressioni dobbiamo verificare se l'utente ha scelto la prima opzione dall'elenco (rappresentata dal valore 1, come visto nel punto 5). Per fare questo, inseriamo la condizione seguente: Maschere![Maschera_Prodotti]![Cornice21] = 1

12. Inseriamo quindi l'azione da eseguire quando la condizione è verificata (Then). Per fare questo, selezioniamo FinestraMessaggio dalla casella a tendina posta accanto al segno di addizione ("+" colorato di verde), che mostrerà a video una finestra di informazioni.

13. Digitiamo nella casella Messaggio il testo che apparirà quando viene selezionato il primo pulsante della casella di opzioni. In questo caso, scriviamo il codice del corriere indicato.

14. A questo punto, dobbiamo ripetere la stessa operazione aggiungendo le condizioni necessarie affinché vengano mostrati a video anche i messaggi per le altre opzioni del gruppo pulsanti, cioè la numero 2, la numero 3 e la numero 4, quindi salvare la macro. Al termine, otteniamo il risultato mostrato a lato.

15. Adesso non ci resta che associare al controllo l'evento che manderà in esecuzione la nostra macro (salvata in precedenza con il nome Macro2), utilizzando l'evento Dopo aggiornamento.

16. Infine, dopo aver salvato la maschera, verifichiamone il funzionamento. Come possiamo notare, quando l'utente seleziona una voce dell'elenco appare una finestra che mostra il codice identificativo del corriere prescelto.

VERIFICA... le competenze

ESERCIZI

1 Apri il file Rappresentanti.accdb

Il database è formato dalla tabella Agenti, la cui struttura è la seguente:

Agenti (ID_Agente, Nome, Agenzia, Indirizzo, Città, Tot_ordini, Fatturato, Provvigione%, Indirizzo_Banca, Sede)

- Crea una maschera collocandovi tutti i campi. Aggiungi una casella di testo di nome Valore e un pulsante di comando che permetta di ottenere l'elenco degli agenti (campi Nome e Città) che hanno un fatturato superiore a quello indicato nella casella di testo Valore. Per fare questo, devi realizzare una query di selezione degli agenti, inserendo nel criterio l'espressione: > [Maschere]![Maschera_Valore]![Valore], dove Valore è il nome della casella di testo aggiunta alla maschera chiamata Maschera_Valore.
- Crea una maschera collocandovi tutti i campi. Aggiungi una casella di testo (citta) e un pulsante di comando. Al clic sul pulsante di comando devi richiamare una query di raggruppamento che calcoli la media del fatturato per la città inserita nel controllo città.
- Confronta la tua soluzione con quella presente nel file Rappresentanti_solux.accdb.

2 Apri il file Cambiali.accdb
- Crea una maschera collocandovi tutti i campi della tabella Cambiali. Aggiungi una casella di testo di nome banca e un pulsante di comando che permetta di ottenere l'elenco delle cambiali (Debitore, Importo, Data_presentazione, Data_scadenza) in base alla piazza/banca indicata nella casella banca. Per fare questo devi realizzare una query di selezione, inserendo nel criterio l'espressione corretta.
- Crea una maschera collocandovi tutti i campi della tabella Ordinativi. Aggiungi un pulsante di comando che al clic effettui il calcolo, tramite una query di aggiornamento, dei giorni (campo gg) e dello sconto (campo Sconto).

3 Apri il file Ordinativi.accdb
- Crea una maschera collocandovi tutti i campi della tabella Ordinativi. Aggiungi una casella di testo di nome cliente e un pulsante di comando che permetta di ottenere l'elenco degli ordini (Data_ordine e Importo) effettuati dal cliente indicato nella casella cliente. Per fare questo devi realizzare una query di selezione degli ordini, inserendo nel criterio l'espressione corretta.
- Crea una maschera collocandovi tutti i campi della tabella Ordinativi. Aggiungi una casella di testo (data) e un pulsante di comando. Al clic sul pulsante di comando devi richiamare una query di raggruppamento che calcoli il massimo e minimo importo degli ordini per la data inserita nel controllo data.

4 Crea il file Agenda.accdb
- Crea una maschera collocandovi tutti i campi della tabella Agenda. Aggiungi un pulsante di comando che permetta di ottenere l'elenco dei recapiti (Tipo_recapito, Numero) in base al nominato selezionato (campo ID_agenda). Per fare questo devi realizzare una query di selezione, inserendo nel criterio l'espressione corretta.
- Crea una maschera collocandovi tutti i campi della tabella Agenda. Aggiungi una casella di testo di nome Provincia e un pulsante di comando che, al clic, calcoli l'altezza e il peso massimi e minimi, tramite una query di raggruppamento, dei nominativi della provincia indicata nella casella Provincia.

VERIFICA... le competenze

ESERCIZI

1 Crea un database di nome video e memorizzalo nella cartella Documenti.
- Crea una tabella, scegliendo opportunamente il tipo, la dimensione e il formato dei campi elencati di seguito:
  - titolo
  - regista
  - attore protagonista
  - genere
  - anno
  - durata (in minuti)
- Per il campo genere assegna una casella combinata che contenga le seguenti voci: "Thriller, Comico, Commedia, Drammatico, Fantascienza, Horror, Sentimentale, Documentario".
- Chiudi e salva la tabella con il nome Film.
- Aggiungi alla tabella Film un campo di nome ID e di tipo contatore.
- Aggiungi il campo in prestito che visualizzerà l'informazione che dice se il film è stato prestato o meno.
- Aggiungi la chiave primaria al campo ID.
- Inserisci almeno 10 record.

2 Crea un database denominato registrazioni e memorizzalo nella cartella Documenti.
- Con i dati elencati di seguito, crea una tabella, scegliendo opportunamente il tipo, la dimensione e il formato dei campi:
  - titolo disco
  - nome cantante
  - anno pubblicazione
  - genere musicale
  - numero brani contenuti
  - prezzo
- Per il campo genere musicale assegna una casella combinata che contenga le seguenti voci: "Pop, Rock, Punk, Heavy Metal, Classica, Swing, Jazz, Disco, House, Hardcore, R&B, Soul, Funky, Progressive, Country, Reggae, Rap, Hip Hop".
- Chiudi e salva la tabella con il nome Dischi.
- Aggiungi alla tabella un altro campo di nome ID e di tipo contatore.
- Sostituisci il campo anno pubblicazione con data pubblicazione e modifica il tipo di dati.
- Aggiungi il campo in prestito dove sarà visualizzata l'informazione che dice se il disco è stato prestato o meno.
- Aggiungi la chiave primaria al campo ID.
- Inserisci almeno 20 record a scelta di generi musicali vari.
- Crea le seguenti query di selezione:
  - elenca tutti i record di un determinato genere musicale (a scelta);
  - elenca tutti i record il cui titolo inizia per "R" e "S";
  - elenca tutti i record dei dischi pubblicati tra il 1995 e il 2002;
  - elenca tutti i record di un determinato cantante (a scelta);
  - elenca tutti i record di un determinato genere musicale (a scelta) pubblicati dopo l'anno 2005;
  - elenca tutti i record di un determinato genere musicale (a scelta) pubblicati prima del 2005;
  - elenca tutti i record che costano tra 10 € e 14 €;
  - elenca tutti i record che hanno un numero di brani compreso tra 5 e 10.
- Crea le seguenti query di raggruppamento:
  - conta quanti dischi sono stati pubblicati prima del 2006;
  - conta quanti dischi ci sono, raggruppandoli per genere musicale;
  - conta quanti dischi costano meno di 10 €;
  - calcola il prezzo medio dei dischi raggruppati per cantante;
  - calcola il numero medio di brani per ogni disco, raggruppati per cantante;
  - calcola il prezzo minimo per ogni disco, raggruppati per cantante;
  - calcola il prezzo massimo per ogni disco, raggruppati per genere musicale;
  - calcola il numero minimo e massimo di brani per ogni disco, raggruppati per genere musicale.

VERIFICA... le competenze

ESERCIZI

3 Crea un database di nome Aerei e memorizzalo nella cartella Documenti.

– Crea le seguenti tabelle, scegliendo opportunamente il tipo, la dimensione e il formato dei campi elencati tra parentesi dopo il nome delle tabelle (in grassetto i campi di tipo chiave primaria):
Aeroporti (**ID_Aeroporto**, Città, Nazione, Num_piste)
Voli (**ID_Volo**, Giorno, Ora_par, id_aeroporto_part, Ora_dest, id_aeroporto_dest, id_aereo)
Aerei (**ID_Aereo**, Num_passeggeri, Qta_merci, Tipo)

– Aggiungi i vincoli relazionali opportuni, come indicato nella seguente figura:

Il campo Tipo indica se è un volo "Cargo" oppure "Passeggeri", mentre il campo Qta_merci è espressa in quintali (q). Il campo Giorno è di tipo testo ("Lun","Mar","Mer", ecc.).

– Esegui le seguenti query di selezione:
  - elenca tutti gli aeroporti che hanno destinazione "Roma";
  - elenca le nazioni di partenza e di arrivo del volo "XC101";
  - elenca gli aerei che trasportano almeno 120 passeggeri e che partono da "Londra" e da "Manchester";
  - elenca tutti i voli in partenza da Milano Malpensa tra il 10 e il 20 aprile 2019 per Londra;
  - elenca tutti gli aerei che trasportano almeno 10 q di merci;
  - elenca tutti gli aeroporti che hanno voli con aerei di tipo "Cargo".

– Esegui le seguenti query di raggruppamento:
  - conta quanti voli partono da Roma ogni giorno;
  - calcola la quantità di merci media per aereo, solo per voli di tipo "Cargo";
  - calcola il numero di piste totali presenti per nazione;
  - conta quanti voli partono da ciascun aeroporto, solo se maggiori di 2;
  - calcola il numero massimo di passeggeri per voli con destinazione "Londra";
  - calcola il numero di voli che partono tra le ore 15 e 22 dall'Italia;
  - calcola la quantità massima di merci in volo il martedì.

ALTERNANZA SCUOLA-LAVORO

Scheda progetto n. 1

ARCHIVIO 2.0 PER L'ALTERNANZA SCUOLA-LAVORO

PROBLEMA

Durante il periodo di alternanza ti viene richiesto di completare l'archivio dell'alternanza ricevuto dalla Camera di commercio. In tale archivio vanno aggiunte le ore di frequenza al corso "Sicurezza e prevenzione nei luoghi di lavoro", propedeutico all'attività di alternanza stessa.
Inoltre ti viene richiesto di:
- **realizzare fisicamente il database;**
- **popolare le tabelle con i dati delle aziende e degli allievi;**
- **effettuare l'accoppiamento dei ragazzi alle ditte disponibili, stampando quindi l'elenco completo;**
- **effettuare la stampa degli elenchi per le firme di presenze al corso propedeutico;**
- **estrarre l'elenco degli allievi dal database, in modo da presentare un documento per la firma di presenza durante il periodo di alternanza.**

Prerequisiti e obiettivi formativi

- Aver effettuato un periodo di alternanza
- Saper analizzare un problema
- Saper interpretare lo schema logico di un database
- Saper realizzare un database utilizzando un DBMS
- Saper estrarre e scambiare dati da due database diversi
- Saper integrare dati a documenti elettronici

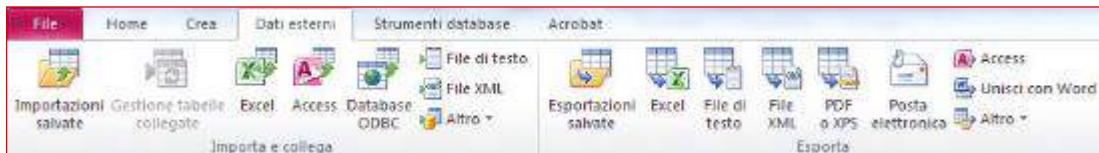
Indicazioni per la progettazione

Il problema da risolvere può essere scomposto in fasi:
a realizzazione del database in un DBMS;
b importazione dei dati delle ditte;
c data entry con i dati degli alunni;
d predisposizione degli elenchi;
e predisposizione di modelli e loro connessone ai dati.

La prima fase richiede di avere a disposizione lo schema logico del database progettato in una precedente attività: partendo da esso si procede alla creazione del database e delle tabella.

Per le due fasi successive è necessario avere a disposizione sia i dati degli alunni che quelli delle ditte: è in ogni caso possibile procedere con la realizzazione del progetto inserendo anche solo alcuni dati di prova per poter effettuare i test ed il collaudo.

Traccia per la realizzazione

1 Crea un nuovo database di Access salvandolo con il nome Alternanza.accdb, quindi definisci la struttura delle tabelle descritte nello schema a lato.

ALTERNANZA SCUOLA-LAVORO

2 Completa la definizione dei campi degli alunni assegnando il tipo corretto per la memorizzazione dei dati richiesti. Per fare questo, basati sui dati necessari all'iscrizione ai corsi di formazione sulla prevenzione e sicurezza nel posto di lavoro.

Se necessario, aggiungi allo schema logico altre tabelle non indicate per completarlo con tutti i dati richiesti.

3 Adesso devi effettuare l'importazione dei dati delle aziende: nell'ambiente di lavoro Access esistono diversi modi per eseguire quanto indicato. La scheda Dati esterni consente di importare ed esportare le tabelle in diversi formati, come mostrato nella figura seguente.

Nel gruppo Importa e collega troviamo le icone che consentono l'importazione in Access di file da diversi formati, mentre nel gruppo Esporta troviamo le icone che consentono di esportare le tabelle in file di diverso formato.

4 Dopo aver individuato il formato necessario ed esserti accertato che sia incluso tra i formati disponibili, effettua l'importazione diretta selezionando il file dal computer. In questo caso, abbiamo selezionato il formato File di testo.

Qualora i nominativi degli alunni non siano già memorizzati in un file, bisogna inserirli manualmente da tastiera, effettuando un'operazione chiamata data entry.

5 Predisponi quindi una maschera per l'input e la modifica dei dati, quindi un report per generare i tabulati di stampa.

6 Infine predisponi moduli in word ed effettua la connessione mediante mail-merge con la base di dati che hai appena realizzato.

Preparazione e presentazione dei risultati

La soluzione di questo problema prevede la realizzazione del database e degli strumenti per popolare le tabelle con i dati e la produzione dei relativi tabulati e documenti necessari per la gestione degli accoppiamenti tra aziende e alunni: la presentazione dei risultati del lavoro consiste quindi nella predisposizione e stampa dei tabulati e nella realizzazione del file .doc contenente tutta la modulistica pronta per essere stampata.

ALTERNANZA SCUOLA-LAVORO

Proposta operativa n. 1

GESTIONE MERCI SUDDIVISE IN SEDI DIVERSE

PROBLEMA

Il tuo tutor aziendale ti chiede di progettare un database per la gestione dei magazzini di un'azienda con più sedi. Le merci sono conservate in magazzini, e in ciascuno di essi è presente una sola categoria merceologica.

Passi operativi

- Crea la struttura della base di dati.
- Individua la struttura delle tabelle coinvolte.
- Crea una maschera per l'immissione dei dati nelle tabelle.
- Crea una maschera con sottomaschera per la visualizzazione dei prodotti per magazzino.
- Definisci le seguenti interrogazioni per ottenere:
  - un elenco di eventuali prodotti presenti in più di un magazzino;
  - un elenco degli articoli che hanno subito operazioni di scarico in un determinato mese;
  - l'elenco dei movimenti di ciascun articolo nell'ultimo anno.
- Crea un prospetto dei prodotti suddivisi per magazzino in grado di calcolare alcuni dati statistici (media, somma, conteggio, minimo ecc.).

Proposta operativa n. 2

STUDIO DI AMMINISTRAZIONE CONDOMINIALE

PROBLEMA

Il tuo tutor aziendale, che dirige uno studio di amministrazione di condomini, ti chiede di realizzare una semplice procedura per la gestione di alcuni dati relativi ai fornitori e alle prestazioni a loro connesse (acquisti di materiale, solleciti, gestione raccomandate, richieste di riparazioni urgenti, ecc.). Lo scopo è quello di avviare un processo di informatizzazione dello studio per produrre le stampe della situazione contabile di ciascun condominio e fornitore. Per realizzare il database dovrai collaborare con la segretaria dello studio che potrà fornirti tutti i dati necessari.

Passi operativi

- Crea la struttura della base di dati.
- Individua la struttura delle tabelle coinvolte.
- Crea una maschera per l'immissione dei dati nelle tabelle.
- Crea una maschera con sottomaschera per la visualizzazione dei fornitori e delle relative riparazioni effettuate.
- Definisci le seguenti interrogazioni per ottenere:
  - un elenco di fornitori che devono ancora essere retribuiti;
  - un elenco dei fornitori che non hanno mai ricevuto solleciti di consegna;
  - l'elenco del materiale acquistato nell'ultimo mese.
- Crea un prospetto del materiale acquistato dai diversi fornitori, suddiviso per mese, in grado di calcolare alcuni dati statistici (media, somma, conteggio, minimo ecc.).

CLIL

Key concepts

Data Base Management System (DBMS)

A database, or data base, is a collection of a large amount of data stored for a long period of time, managed by a type of software called DBMS (Data Base Management System).
Access is a very popular DBMS; files saved with Access have the extension .accdb.

Access databases contain several objects: tables, queries, masks, reports, pages, macros.
Tables have a structure consisting of rows, called records, and columns, called fields.

Tables are used to create queries, masks, and reports.

Queries allow users to query the database, that is, to search information contained within one or more tables based on certain search criteria. A query retrieves data to be displayed, printed, deleted, or entered in a new table. Both masks and reports can be created from queries.

Relationships connect different tables to one another. The one-to-many relationship connects two tables, called the referencing and the referred table.
A relationship binds two fields called primary key (in the referred table) and foreign key (in the referencing table).

The primary key of a table is a field that identifies unequivocally each record.

Primary key fields have unique values, that is, no repetitions are allowed throughout a table.

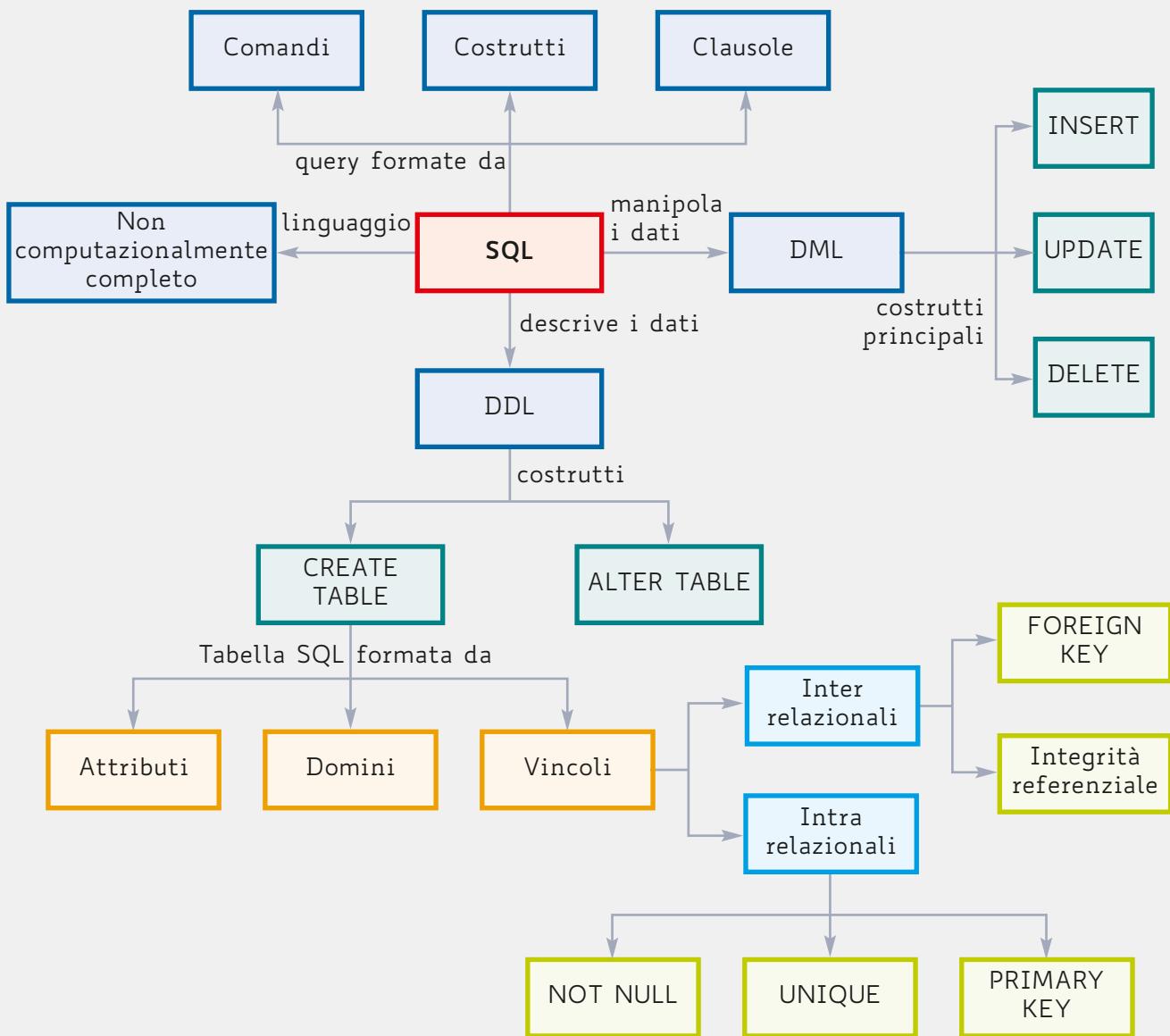
Referential integrity ensures that all the relationships between related tables are valid and that no accidental variation or data cancellation can happen.

Quiz

1 Which of the following is not an Access table field type?
a Number
b Text
c Boolean
d Sound

2 Microsoft Access belongs to the software category:
a Spreadsheets
b Database
c Word Processor
d Operating systems

3 What is a record of a database?
a The column of a table
b A group of tables
c A relationship between two masks
d The row of a table

Il linguaggio SQL

CONOSCENZE

- Riconoscere le caratteristiche di DDL, DML e QL
- Identificare i principali comandi SQL
- Comprendere il ruolo del linguaggio SQL
- Individuare i principali elementi dei comandi SQL
- Conoscere la struttura dei comandi SQL
- Comprendere il significato di comando, clausola e costrutto

COMPETENZE

- Saper interrogare il database attraverso query di selezione
- Realizzare query contenenti congiunzioni tra tabelle
- Realizzare query con operatori aggregati
- Applicare gli operatori relazionali alle query SQL

ABILITÀ

- Applicare i comandi SQL
- Utilizzare gli operatori di aggregazione
- Creare query complesse
- Creare query con congiunzioni multiple
- Creare query annidate

Il file relativi a esempi, esercizi e a eventuali soluzioni di questa unità sono contenuti nel CD-ROM e sono anche scaricabili dal sito hoepliscuola.it

I linguaggi DDL e DML

IN QUESTA LEZIONE IMPAREREMO...

- ad applicare i comandi SQL di creazione tabella
- a riconoscere e applicare i costrutti DML
- a riconoscere e applicare i costrutti DDL

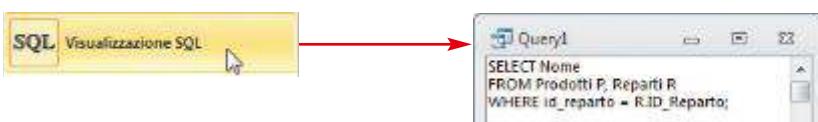
MAPPA CONCETTUALE

didattica inclusiva

Il linguaggio SQL

Il linguaggio SQL (*Structured Query Language*, "linguaggio di interrogazione strutturato") consente di interagire con i dati presenti nelle basi di dati.

Ricordiamo che le basi di dati sono formate dalle seguenti componenti:
– schema (struttura), cioè i campi che costituiscono la tabella;
– istanza, cioè il record della tabella;
– tupla, cioè la riga della tabella (a volte il risultato di un'interrogazione).

Computazionalmente completo
La complessità computazionale studia le risorse minime necessarie, in termini di tempo e di calcolo delle risorse, per la risoluzione di un problema. Un linguaggio non computazionalmente completo non è in grado di risolvere problemi di varia natura bensì specifici.

Se utilizzato da solo, tuttavia, l'SQL non permette di programmare un'intera applicazione, in quanto non è **computazionalmente completo**.
Nella maggior parte dei DBMS il linguaggio SQL viene usato per consentire l'accesso ai dati. SQL può essere suddiviso in quattro sottolinguaggi:
– DDL (*Data Definition Language*). Appartengono a questa sezione di SQL tutti i comandi preposti alla creazione di tabelle e alla definizione appunto dei dati in esse contenuti. Mediante questo linguaggio si definiscono le strutture delle relazioni del database;
– DML (*Data Manipulation Language*). Appartengono a questa sezione di SQL tutti i comandi per la modifica dei dati contenuti nelle tabelle, in termini di aggiunta, eliminazione, modifica, aggiornamento;
– DCL (*Data Control Language*). Appartengono a questa sezione di SQL tutti i comandi che rendono possibile la gestione dei permessi di accesso alle risorse del database;
– QL (*Query Language*). Questa sezione rappresenta il fulcro di quasi tutte le attività legate ai database, in quanto i comandi appartenenti a questa sezione di SQL consentono l'interrogazione, il raggruppamento, il conteggio e l'ottenimento di prospetti personalizzati dei dati presenti nelle varie tabelle dei database.

In Access, per scrivere il codice in linguaggio SQL, dopo essere entrati nella struttura della Query, è necessario attivare il menu denominato Visualizzazione SQL.

Per verificare l'esecuzione del codice SQL digitato è sufficiente fare clic sul pulsante Esegui.
In seguito all'attivazione di questo comando si apre una finestra di dialogo che mostra il risultato ottenuto dall'esecuzione della query.

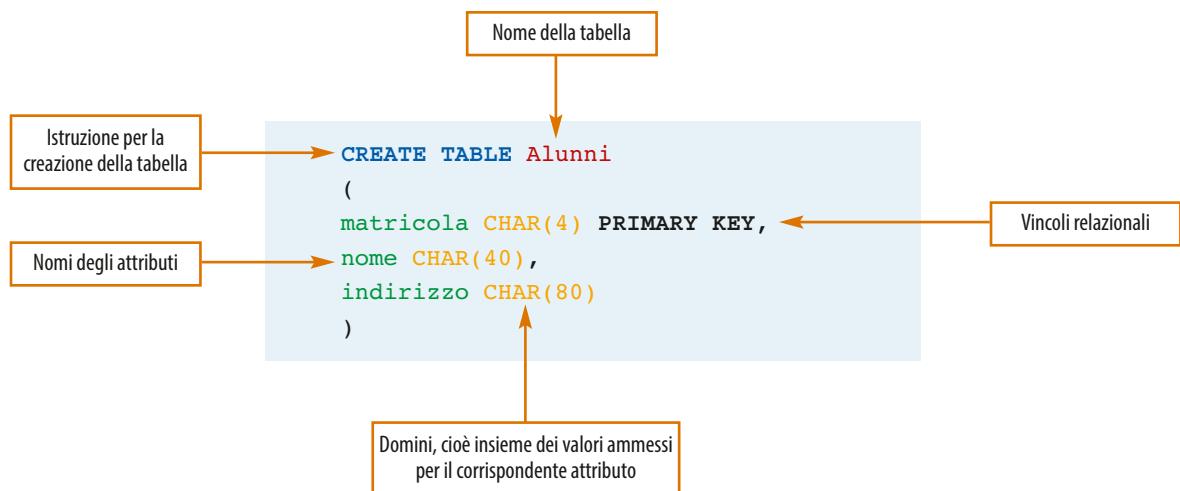
Ciascuna query contiene codice SQL ed è suddivisa in:
– comando;
– costrutto;
– clausole.

Il comando è l'intero blocco di codice che termina con il punto e virgola (;), il costrutto è invece la tipologia di query effettuata, come per esempio SELECT, INSERT, CREATE ecc. Le clausole, infine, sono i parametri aggiuntivi di un costrutto, come per esempio FROM, WHERE, IN, ON ecc. Secondo lo standard è più leggibile un codice con clausole e costrutti digitati in maiuscolo, nonostante il linguaggio non sia case sensitive, e il resto della stringa in minuscolo (campi, operatori relazionali, operatori logici ecc.).

La creazione delle tabelle in SQL

Una tabella in SQL, che viene creata attraverso il sottolinguaggio DDL, è per definizione composta dai seguenti tipi di informazioni:
– attributi;
– domini, di tipo standard;
– vincoli intra relazionali (opzionali).

Il costrutto di creazione di una tabella è CREATE TABLE, che ha la seguente sintassi:

```
CREATE TABLE nome tabella(
nome_campo tipo [(lunghezza)]
[NOT NULL] [PRIMARY KEY] [UNIQUE]}[,...]
[PRIMARY KEY (nome campo, ...)]
[UNIQUE [nome indice] ({nome campo}[,...])]
[CONSTRAINT [nome vincolo] REFERENCES nome tabella(nome campo)])
```

ESEMPIO

CREARE UNA NUOVA TABELLA

In questo esempio vogliamo creare una tabella per memorizzare tutti i dati degli alunni di una scuola. Gli attributi sono il nome e il cognome dell'alunno (nome), la matricola e l'indirizzo. Il codice SQL necessario alla creazione di tale tabella è il seguente:

```
CREATE TABLE Alunni (
 matricola CHAR(4) PRIMARY KEY,
 nome CHAR(40),
 indirizzo CHAR(80) );
```

Vediamo il significato del comando SQL.

Le tabelle sono formate da campi (colonne) e record (righe), seguendo lo schema di qualunque altro database. I campi possiedono un nome (attributo) e un tipo (dominio), che devono essere decisi in fase di creazione delle tabelle. La tabella che segue riporta principali nomi e tipi di campo sintetizzandone il significato.

Tipo di campo	Nome campo	Significato
Numerico	INT	Intero
	FLOAT	Reale
	CURRENCY	Valuta
Testo	CHAR	La dimensione è fissa. Per esempio, se memorizziamo la stringa 'esempio' (lunga 7 caratteri) in un campo definito come CHAR(20), andiamo a occupare 20 caratteri, e ai 7 caratteri della parola verranno aggiunti 13 spazi.
	VARCHAR	La dimensione è variabile. Per esempio, se memorizziamo la stringa 'esempio' (lunga 7 caratteri) in un campo definito come VARCHAR(20), andiamo a occupare solo i 7 caratteri necessari.
	NOTE	Viene anche chiamato campo MEMO e consente di inserire fino a 65536 caratteri.
Contatori	AUTOINCREMENT	Auto incrementante
Data	DATE	Data in formato gg/mm/aa

I vincoli intra relazionali verificano l'input dei dati di una tabella; se i dati corrispondono allo schema di immissione vengono accettati, altrimenti no. I vincoli intra relazionali sono condizioni che coinvolgono una sola relazione (distinguibile ulteriormente a livello di tupla o di tabella): li descriviamo di seguito.

- NOT NULL: vale per l'attributo in cui è indicato, significa che l'attributo non può essere lasciato vuoto.
- UNIQUE: può essere definito su un solo campo, indicandolo subito dopo il nome e il tipo di campo.
- PRIMARY KEY: definisce uno o più campi i cui valori identificano in modo univoco ciascun record della tabella. Indica una chiave primaria univoca per tutta la tabella, che deve sempre essere di tipo NOT NULL.

ESEMPIO

CREARE UNA TABELLA CON VINCOLI INTRA RELAZIONALI

In questo esempio vogliamo creare una tabella per la memorizzazione dei dati relativi alle materie di studio di una scuola, dei relativi voti ottenuti dagli alunni in ciascuna di esse e dei dati anagrafici degli insegnanti. Iniziamo con il creare la tabella delle Materie, dove si individua, come vincolo intra relazionale, la chiave primaria (PRIMARY KEY) per il campo ID_materia:

```
CREATE TABLE Materie (
 ID_materia INT PRIMARY KEY,
 materia CHAR(20) NOT NULL );
```

Quindi creiamo la tabella dei Voti, dove il vincolo intra relazionale è rappresentato dalla chiave primaria (PRIMARY KEY) del campo ID_voto:

```
CREATE TABLE Voti (
 ID_voto INT PRIMARY KEY
 data date NOT NULL,
 id_materia INT NOT NULL,
 id_alunno INT NOT NULL,
 voto INT NOT NULL,
 );
```

Creiamo infine la tabella dei Docenti, dove il vincolo intra relazionale è rappresentato dalla chiave primaria (PRIMARY KEY) del campo ID_docente:

```
CREATE TABLE Docenti (
 ID_docente INT PRIMARY KEY,
 cognome CHAR(25) NOT NULL,
 nome CHAR(25) NOT NULL,
 telefono CHAR(12) );
```

Vincolo inter relazionale
Il vincolo inter relazionale rappresenta un vincolo posto tra dati presenti in due tabelle diverse; si tratta di una chiave primaria (PRIMARY KEY) e di una chiave esterna (FOREIGN KEY).

Integrità referenziale
Per integrità referenziale si intende l'insieme dei vincoli inter relazionali tesi ad assicurare che le relazioni tra i record delle tabelle correlate siano valide e che i dati tra loro collegati non vengano eliminati o modificati per errore.

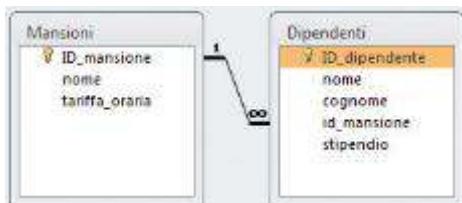
I vincoli inter relazionali

I **vincoli inter relazionali** permettono di creare un legame permanente tra diverse tabelle.
Per relazionare le tabelle Voti e Alunni in modo tale che a ogni record immesso nella tabella Voti (che rappresenta una singola valutazione) corrisponda un allievo della tabella Alunni, dovremo usare un vincolo inter relazionale che verifichi la presenza nella tabella Alunni dell'alunno al quale è associato il voto stesso. Ciò consente di verificare l'esistenza di un elemento della tabella correlata e prende il nome di vincolo di **integrità referenziale**.

Vediamo adesso la sintassi SQL che consente di realizzare un vincolo inter relazionale tra la chiave esterna di una tabella e la chiave primaria, presente in una seconda tabella indicata nel codice con il nome Tabella 2:

```
campo chiave esterna CONSTRAINT nome vincolo REFERENCES Tabella2(campo
chiave primaria)
```

ESEMPIO

VINCOLO INTER RELAZIONALE
Vogliamo memorizzare i dati relativi ai dipendenti di una azienda e alle mansioni svolte da ciascuno di essi. La situazione è quella in cui più dipendenti possono svolgere la stessa mansione, che tra l'altro ha un tariffario ben preciso. Potremmo risolvere il problema mediante due tabelle, la prima chiamata Mansioni e la seconda Dipendenti.

```
CREATE TABLE Mansioni (
 ID_mansione AUTOINCREMENT PRIMARY KEY,
 nome CHAR(20) NOT NULL,
 tariffa_oraria INT);

CREATE TABLE Dipendenti (
 ID_dipendente INT(4) PRIMARY KEY,
 nome CHAR(20) NOT NULL,
 cognome CHAR(20) NOT NULL,
   id_mansione INT NOT NULL CONSTRAINT m REFERENCES Mansioni(ID_mansione),
   stipendio CURRENCY);
```

Vincolo inter relazionale tra il campo id_mansione, della tabella Dipendenti, e la chiave primaria ID_mansione, della tabella Mansioni

In questo caso sono state definite due tabelle poste in relazione tra loro. La tabella Mansioni è in relazione con Dipendenti secondo il campo indicato (ID_mansione con id_mansione). Tale vincolo re-

lazionale permette una verifica istantanea a ogni immissione o modifica di record all'interno delle due tabelle, ed è chiamato relazione 1 a n (in Access 1 a... infinito).

La modifica della tabella

La struttura di una tabella può essere modificata mediante il costrutto ALTER, che consente la variazione degli attributi, del dominio, degli indici e dei vincoli. La sintassi prevede sempre una chiave primaria come indice, alla quale viene assegnato un nome. Per cancellare una chiave primaria si deve sempre fare riferimento al nome associato all'indice. Inoltre, non è possibile rinominare un campo mantenendone inalterato il contenuto. La sintassi di modifica di una tabella è riportata di seguito:

```
ALTER TABLE nome tabella { ADD | ALTER | DROP }
[COLUMN tipo di campo [(lunghezza)] [CONSTRAINT nome indice] |
COLUMN tipo di campo [(lunghezza)] |
[CONSTRAINT indicemulticampo]
```

La modifica dei domini è subordinata alla compatibilità rispetto al contenuto della tabella. Se, per esempio, un campo di CHAR viene modificato in INTEGER è chiaro che solo i valori numerici verranno trasformati. Per gli interventi sulla lunghezza, nei campi testo, se la dimensione diminuisce, i caratteri in eccedenza vengono troncati.

Indice
Un indice consente di scansionare i record della tabella in maniera più veloce. Un indice non è altro che una tabella (invisibile all'utente) che contiene le posizioni dei record secondo un determinato ordine, che può essere crescente o decrescente. Durante una query i record verranno scanditi leggendone la posizione dall'indice. In generale, conviene associare un indice a un campo che verrà utilizzato per effettuare ricerche (SELECT).

Prendiamo per esempio in esame le tabelle Mansioni e Dipendenti dell'esempio precedente. Il codice che segue aggiunge un nuovo campo di nome data alla tabella Mansioni:

```
ALTER TABLE Mansioni ADD COLUMN data DATE;
```

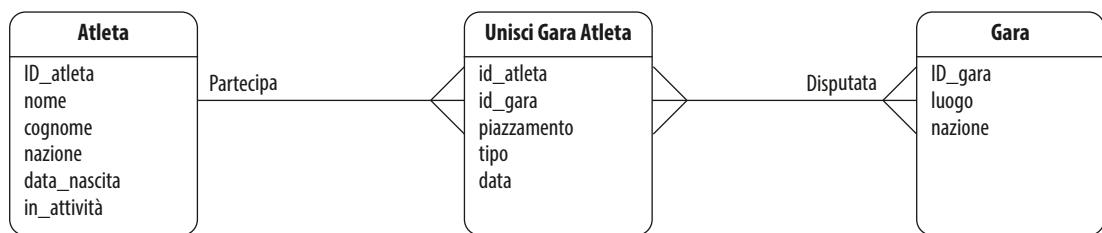
Per eliminare il campo data utilizziamo la clausola DROP COLUMN:

```
ALTER TABLE Mansioni DROP COLUMN data;
```

Per creare un **indice** nella tabella Dipendenti, che chiameremo cog_nom, basato sui campi cognome e nome, utilizziamo la clausola ADD INDEX:

```
ALTER TABLE Dipendenti ADD INDEX cog_nom(cognome, nome);
```

Per eliminare una intera tabella si utilizza DROP TABLE. Per esempio, per cancellare la tabella Mansioni, la sintassi è:

```
DROP TABLE Mansioni;
```

Per cambiare il nome ma non il contenuto di una tabella, in questo caso della tabella Dipendenti in Subalterni, si utilizza il comando RENAME:

```
ALTER TABLE Dipendenti RENAME Subalterni;
```

ESEMPIO

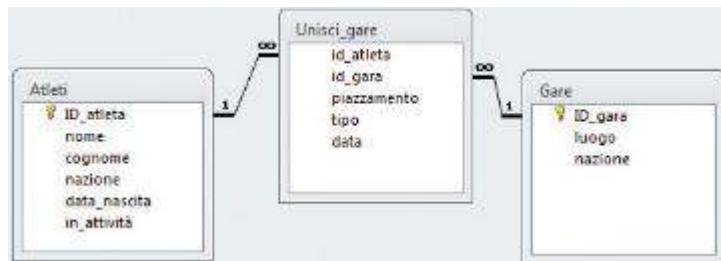
GESTIONE GARE ATLETI

In questo esempio vogliamo organizzare un database, realizzando le opportune tabelle e relazioni, per la memorizzazione dei dati relativi agli sciatori e alle gare a cui partecipano. Prima di tutto occorre memorizzare i dati anagrafici dei vari sciatori iscritti, tenendo presente che ciascuno sciatore gareggia per una certa nazione. Si deve quindi memorizzare, per ogni gara disputata, il piazzamento e il luogo in cui si è gareggiato. Per gestire il database indicato decidiamo di creare tre tabelle, in relazione tra loro, in modo da gestire per ogni sciatore più gare effettuate e per ogni gara i piazzamenti dei vari atleti. La tabella Atleti, contenente i dati anagrafici degli sciatori, possiede una chiave primaria rappresentata dall'identificativo dello sciatore (ID_atleta). La tabella Gare contiene i dati della singola gara del campionato. Poiché tra Gare e Atleti esiste una relazione (n, n), viene definita la tabella Unisci_gare, che contiene il tipo di gara effettuata (slalom, combinata ecc.), la data in cui è stata effettuata e il piazzamento, secondo il seguente modello E-R:

La codifica in SQL della definizione della tabella è la seguente:

```
CREATE TABLE Atleti (
 ID_atleta AUTOINCREMENT PRIMARY KEY,
 nome CHAR(20),
 cognome CHAR(20),
 nazione CHAR(20),
 data_nascita DATE,
 in_attività CHAR(1));

CREATE TABLE Gare (
 ID_gara INT NOT NULL PRIMARY KEY,
 luogo CHAR(20),
 nazione CHAR(20));

CREATE TABLE Unisci_gare (
 id_atleta INT NOT NULL CONSTRAINT M REFERENCES Atleti(ID_atleta)
 id_gara INT NOT NULL CONSTRAINT M REFERENCES Gare(ID_gara)
 piazzamento INT NOT NULL,
 tipo CHAR(10),
 data DATE NOT NULL);
```

In Access, ciascuna query SQL deve essere salvata singolarmente.

L'ordine di esecuzione delle query deve tenere conto delle regole di visibilità: si deve sempre iniziare da una tabella che non contiene vincoli inter relazionali, in questo caso Atleti, oppure Gare. La tabella Unisci_gare dovrà essere creata necessariamente per ultima.

L'esecuzione delle query di creazione dà luogo alla seguente struttura relazionale:

Il linguaggio DML

Il linguaggio DML (*Data Manipulation Language*) permette l'inserimento e l'eliminazione dei record e l'aggiornamento dei dati contenuti nelle tabelle attraverso i tre seguenti costrutti:
– INSERT;
– DELETE;
– UPDATE.

Il costrutto INSERT

Il costrutto INSERT consente di inserire nuove righe o record in una tabella. Il nuovo record viene immesso in coda, sempre che non vìoli i vincoli inter relazionali, intra relazionali, oppure di integrità referenziale. La sintassi generale di questo costrutto è la seguente:

```
INSERT INTO tabella (campo1, campo2, ... campo n)
VALUES ('valore1','valore2', ... 'valore n');
```

Vediamo ora una applicazione del costrutto INSERT attraverso l'inserimento di un nuovo record nella tabella Atleti:

```
INSERT INTO Atleti (nome, cognome, nazione, data_nascita, in_attività)
VALUES ("Sofia","Goggia","Italia","15/11/1992",true);
```

Siccome ai campi di tipo numerazione automatica (AUTOINCREMENT) non può essere assegnato alcun valore, il campo ID_atleta è stato escluso dalla lista dei campi posti tra parentesi.

Il costrutto DELETE

L'eliminazione di tuple da una tabella si ottiene mediante il costrutto DELETE, che permette di cancellare definitivamente da una tabella anche più di una riga per volta. La sua sintassi è:

```
DELETE FROM nome tabella WHERE condizione
```

Se non viene immessa nessuna condizione WHERE, vengono cancellati tutti i record della tabella; per esempio, la seguente query SQL cancella tutto il contenuto della tabella Gare:

Operatori logici
Consentono di realizzare condizioni multiple nel costrutto WHERE o HAVING e sono: AND, OR, NOT. Se usiamo l'operatore AND entrambe le condizioni devono essere vere affinché il risultato complessivo sia vero, mentre se usiamo OR è sufficiente che solo una condizione sia vera affinché il risultato complessivo sia vero.

```
DELETE FROM Gare;
```

La seguente query SQL, invece, cancella, nella tabella Atleti, solo i record che soddisfano la condizione di avere cognome uguale a "Janka" e nome uguale a "Carlo":

```
DELETE FROM Atleti WHERE cognome="Janka" AND nome="Carlo";
```

In questo caso abbiamo utilizzato un **operatore logico** (AND) per unire le due condizioni.

Il costrutto UPDATE

Con il costrutto UPDATE si ottiene l'aggiornamento dei dati contenuti nei record di una tabella in base a una eventuale condizione. La sintassi del costrutto UPDATE è la seguente:

```
UPDATE tabella SET Attributo = valore o espressione WHERE condizione;
```

Per esempio, nella tabella Atleti, per aggiornare il campo in attività per il record dello sciatore "Blardone", la sintassi sarà la seguente:

```
UPDATE Atleti SET in_attività = false WHERE cognome="Blardone";
```

Se la condizione WHERE non viene inserita, la modifica avverrà per tutti i record della tabella.

ESEMPIO

GESTIONE PELLICOLE CINEMATOGRAFICHE
Apriamo il database Film.mdb formato da una sola tabella chiamata Film, che contiene i dati di alcuni film. In base a una chiave primaria rappresentata dal codice della pellicola, si memorizzano i dati relativi allo sceneggiatore, al regista, al titolo, al costo, agli incassi, al fotografo e alla data di uscita. Vediamo alcune query.

1 Inseriamo un nuovo record nella tabella Film alla posizione 12:

```
INSERT INTO Film VALUES
(12, "Terry Hayes", "Albert Hughes", "La vera storia di Jack lo
squartatore", 120, 210, "Peter Deming", 01/01/2001);
```

2 Aggiungiamo alla struttura della tabella il campo Produttore di tipo char:

```
ALTER TABLE Film
ADD Produttore CHAR(20);
```

3 Aggiorniamo tutti i record riducendo l'incasso del 20%:

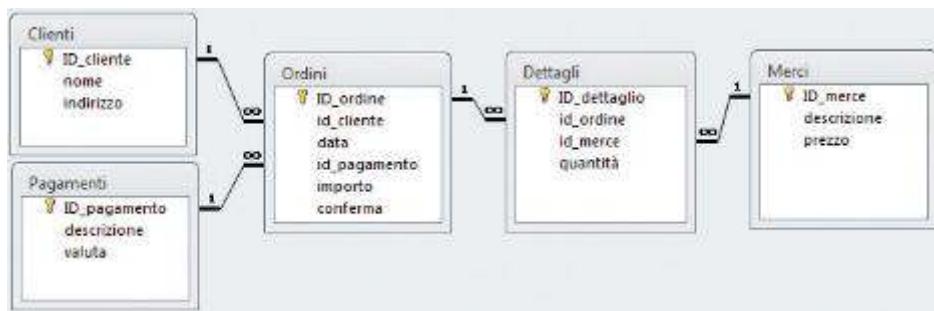
```
UPDATE Film
SET incassi = incassi*0,8;
```

4 Cancelliamo tutti i film che hanno un costo inferiore a 100:

```
DELETE FROM Film WHERE costo < 100;
```

VERIFICA... le competenze

AREA DIGITALE

ESERCIZI

1 Crea un database Libri. Crea le tabelle con i tipi di campi e i relativi vincoli relazionali che ritieni più idonei a rappresentare la situazione riprodotta a lato.

Confronta la tua soluzione con quella contenuta nel database Libri_solux.

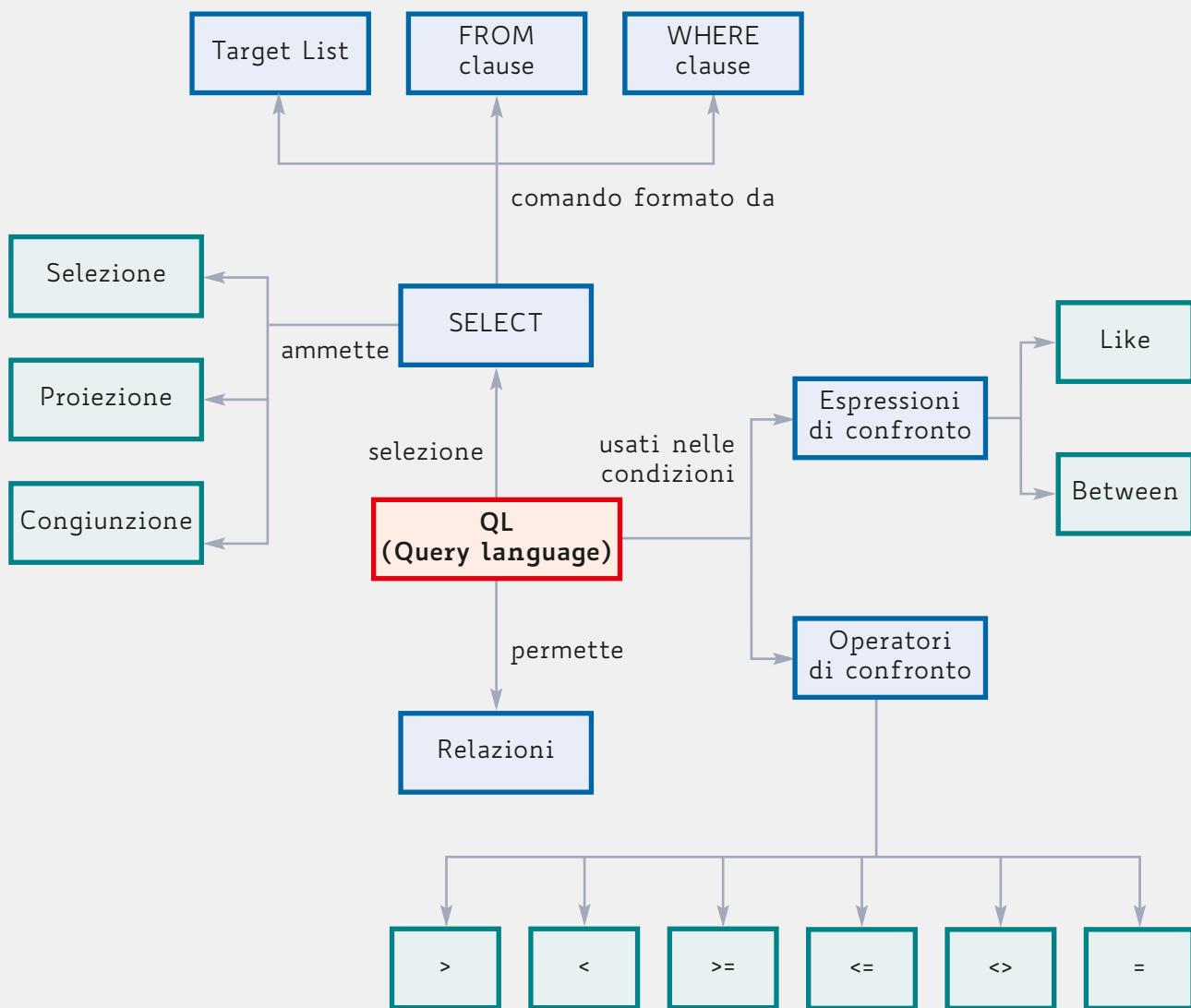
2 Crea un database Gestione_ordini. Crea le tabelle con i tipi di campi e i relativi vincoli relazionali che ritieni più idonei a rappresentare la situazione seguente:

Confronta la tua soluzione con quella contenuta nel database Ordini_solux.

3 Definisci un database mediante codice SQL per memorizzare i dati relativi a una gara podistica. A ciascun concorrente è attribuito un numero di pettorale e viene stabilita la categoria per la quale partecipa (senior, junior ecc.). Se durante la gara commette delle scorrettezze può essere ammonito, alla terza ammonizione scatta la squalifica. Ciascun atleta inoltre gareggia per una determinata squadra. Definisci le relazioni opportune.

4 Definisci un database mediante codice SQL per memorizzare i dati delle gare di Formula Uno degli ultimi anni. Si devono memorizzare le marche delle macchine, le caratteristiche tecniche (per esempio: cilindrata, potenza, velocità massima ecc.), i tipi di pneumatici adottati in quel particolare anno e il nome del pilota sempre relativo a quell'anno. Definisci le relazioni corrette e i tipi di campi corretti di tali tabelle.

5 Definisci un database mediante codice SQL per memorizzare i dati degli abbonamenti di una casa editrice che pubblica più riviste. Ogni rivista ha molti abbonati e un abbonato può sottoscrivere più abbonamenti a riviste diverse. Progetta le relazioni e le tabelle opportune.

6 Definisci un database mediante codice SQL per memorizzare i dati anagrafici delle automobili, degli assicurati e dei sinistri verificatisi nel corso del tempo. Per ogni assicurato possono esserci più sinistri in date diverse o nella stessa data, così come per ogni automobile. Inoltre, per ogni sinistro devi memorizzare la data, il codice delle due assicurazioni (danneggiato e colpevole) e l'importo della pratica di riparazione (eventuale). Definisci la relazione e le strutture dati corrette per memorizzare tale struttura.

Le interrogazioni del database

IN QUESTA LEZIONE IMPAREREMO...

- ad applicare le query d'interrogazione, definendone le relazioni
- a utilizzare selezioni, proiezioni e congiunzioni
- a distinguere il significato degli operatori di confronto

MAPPA CONCETTUALE

Target List

FROM clause

WHERE clause

comando formato da

SELECT

ammette

Selezione

Proiezione

Congiunzione

selezione

QL (Query language)

usati nelle condizioni

Espressioni di confronto

Like

Between

Operatori di confronto

permette

Relazioni

>

<

>=

<=

<>

=

Le interrogazioni SQL

Query Language
Il termine query (richiesta) rappresenta l'azione che viene eseguita sul database, cioè la richiesta di informazioni tramite una operazione di ricerca in base a dei criteri.

Le operazioni di interrogazione del database appartengono al QL (**Query Language**), che permette di operare sulle tabelle effettuando ricerche di informazioni.

L'interrogazione di un database è una delle operazioni che viene effettuata con maggiore frequenza: grazie alle interrogazioni possiamo infatti consultare le tabelle per ottenere dei risultati. L'interrogazione del database è resa possibile dal costrutto SELECT e dalle sue clausole.

Il costrutto SELECT

La sintassi del costrutto SELECT si compone di tre parti principali:
– target list;
– FROM clause;
– WHERE clause.

Nella target list vengono indicati i nomi dei campi (attributi) che verranno estratti dall'interrogazione: devono essere indicati subito dopo la parola chiave SELECT e separati dalla virgola:

```
SELECT A1,…,An          (attributi da includere nel risultato)
```

Nella FROM clause si indicano invece i nomi delle tabelle coinvolte nella query, separate dalla virgola:

```
... FROM tabella1,…,tabellan
```

Nella WHERE clause, infine, si deve indicare la condizione di ricerca:

```
WHERE condizione          (condizione che i dati cercati soddisfano)
```

La sintassi completa del costrutto SELECT è la seguente:

```
SELECT
{* | {nome campo [AS alias]}[,...]}
FROM {nome tabella [[ AS] alias]}[,...]
[WHERE condizione]
[GROUP BY {{nome campo | espressione} [ASC | DESC]}[,...]]
[HAVING condizione]
[ORDER BY {{nome campo | espressione} [ASC | DESC]}[,...]]
```

Sia nella target list che nella clausola FROM possono essere inseriti degli alias. Un alias è una stringa che può essere associata a un campo, oppure a una tabella, mediante il parametro AS. Il codice seguente mostra l'uso di due alias (A e P) rispettivamente per le tabelle Articoli e Percentuali, e un alias ([Percentuale sconto]) per il campo perc.

```
SELECT A.descrizione, P.perc AS [Percentuale sconto]
FROM Articoli AS A, Percentuali AS P
WHERE A.codice = P.codice;
```

Grazie agli alias, otteniamo in generale una maggiore leggibilità della query. In questo caso, la tabella dalla quale provengono i campi descrizione e perc è individuata dagli alias A per Articoli e P per Percentuali. L'alias usato nella target list consente di migliorare la visualizzazione dei risultati della query: in questo caso, l'utente visualizza a video la stringa "Percentuale sconto" al posto di P.perc.

Secondo la sintassi SQL il parametro AS può essere omesso nella clausola FROM, inoltre nella target list gli alias devono essere inseriti tra parentesi quadre.

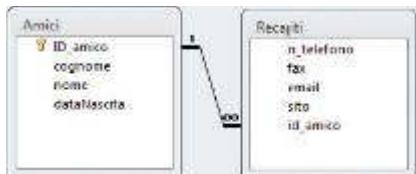
Le query eseguono il prodotto cartesiano delle tabelle indicate nella clausola FROM, considerando le sole righe che soddisfano la condizione posta nella clausola WHERE, per ogni riga ottenuta vengono poi valutate le espressioni presenti nella target list.
Per esempio, per ottenere tutte le descrizioni e l'importo degli articoli che hanno un prezzo inferiore a 5 dalla tabella Articoli del database ArticoliSconti.mdb, indichiamo nella target list i campi descrizione e prezzo, nella FROM clause la tabella Articoli e infine nella WHERE clause la condizione prezzo < 5:

```
SELECT descrizione, prezzo
FROM Articoli
WHERE prezzo <5;
```

Il comando SELECT esprime tre operazioni fondamentali:
– selezione;
– proiezione;
– congiunzione.

La selezione si esprime attraverso la condizione WHERE, che permette di selezionare le tuple della relazione che si vogliono ricercare. La selezione è una sorta di estrazione orizzontale dei record in base a una condizione.
La query seguente mostra una selezione effettuata sulla tabella Amici in base al campo citta:

```
SELECT * FROM Amici WHERE citta="Napoli";
```

Questa query seleziona dunque dalla tabella solo i nominativi di coloro che abitano a Napoli.

In una tabella, se vogliamo selezionare tutti i campi, possiamo sostituirli con il simbolo asterisco (*): in questo caso, si parla di selezione senza proiezione.

La proiezione si esprime invece attraverso i campi che si vogliono selezionare; si tratta quindi di un'estrazione verticale della tabella, che definisce le colonne che deve avere la relazione risultante dalla query:

```
SELECT cognome, nome FROM Amici WHERE citta="Napoli";
```

Questa query estrae quindi dai record ottenuti dalla selezione (WHERE citta="Napoli") solo i campi cognome e nome.

La congiunzione, infine, esprime una relazione fra tabelle e può essere ottenuta mediante appositi costrutti, detti JOIN, oppure mediante la clausola WHERE seguita dai campi oggetto di relazione.
Per esempio, in presenza di due tabelle, nella prima delle quali vi sono i nostri amici e nella seconda i relativi recapiti telefonici, colle-

gati dal campo ID della prima tabella, vogliamo visualizzare i vari recapiti telefonici di ogni amico:

```
SELECT A.nome, A.cognome, R.n_telefono
FROM Amici A, Recapiti R
WHERE A.ID_amico = R.id_amico;
```

Analizzando il codice di questa query, notiamo che la condizione WHERE unisce, per così dire, le due tabelle riferendosi a quelle righe che hanno lo stesso valore per il campo ID.

Esaminiamo ora più nel dettaglio nel paragrafo che segue come porre in relazione tra loro più tabelle attraverso il costrutto SELECT.

Dot notation
La dot notation è una notazione utilizzata nella sintassi di numerosi linguaggi di programmazione. Prevede l'uso del punto (.) come separatore tra elementi diversi, quali per esempio gli oggetti, oppure i record. In SQL separa i nomi delle tabelle dai nomi dei campi. Per esempio, Rubrica.cognome rappresenta il campo cognome della tabella Rubrica.

Il costrutto SELECT e le relazioni

Quando vogliamo interrogare più tabelle legate tra loro da una relazione, dobbiamo tenere in considerazione i campi coinvolti nella congiunzione tra le tabelle. Affinché la relazione si realizzi, è necessario che gli attributi vengano collegati attraverso una condizione WHERE, anche se esistono altre tecniche per effettuare una congiunzione, come quella che usa il costrutto chiamato JOIN.

Per evitare confusione, è sempre utile rinominare le tabelle con alias e utilizzare la **dot notation** , indicando i nomi dei campi preceduti dall'alias della tabella che li contiene, come per esempio:

```
SELECT ...
FROM Prodotti P, Fornitori F
... WHERE F.ID_fornitore = P.id_fornitore ...
```

In questo caso, possiamo chiaramente individuare l'origine dei due campi inseriti nella condizione: il primo è contenuto nella tabella Fornitori, il secondo nella tabella Prodotti.

Per rendere più efficace la lettura delle relazioni il nome dei campi di tipo chiave primaria viene fatto precedere dall'identificatore ID in maiuscolo, mentre le chiavi esterne sono precedute dall'identificatore id in minuscolo.

Proponiamo di seguito una serie di esempi di query di selezione, utilizzando due tabelle del database PazientiVisite.mdb che contengono i dati dei pazienti e delle relative visite mediche che hanno effettuato nel corso degli anni: Pazienti e Visite. Come possiamo notare, le due tabelle sono in relazione 1 a n tramite la chiave primaria ID_paziente della tabella Pazienti con id_paziente (chiave esterna) della tabella Visite.

Vediamo il codice SQL necessario a effettuare sulle tabelle le interrogazioni di seguito indicate.

1 Selezione di tutti i dati relativi ai pazienti:

```
SELECT * FROM Pazienti;
```

2 Selezione dei cognomi di tutti i pazienti:

```
SELECT cognome FROM Pazienti;
```

3 Selezione dei cognomi e dei codici ASL dei pazienti nati in provincia di Milano:

```
SELECT cognome, codAsl
FROM Pazienti
WHERE provincia = "MI";
```

4 Selezione dei cognomi dei pazienti con pressione minima più alta di 85 e massima più alta di 125:

```
SELECT P.cognome, V.pressioneMAX, V.pressioneMIN
FROM Pazienti P, Visite V
WHERE V.pressioneMIN>85 AND V.pressioneMAX>125 AND P.ID_paziente =
V.id_paziente;
```

Vediamo adesso un esempio più articolato di query di selezione su più tabelle poste in relazione.

ESEMPIO

QUERY DI SELEZIONE SU PIÙ TABELLE IN RELAZIONE

Vogliamo conoscere la descrizione e la percentuale di aumento di prezzo degli articoli presenti nelle tabelle seguenti:

Tabella **Percentuali**

ID_percentuale	Percentuale
12a	10%
15b	3,5%

Tabella **Articoli**

Descrizione	id_percentuale	Prezzo
penna a sfera	12a	13,20
refill rosso	15b	12,54
matita grafite 0,5 HB	12a	3,96
mina HH 0,5	15b	4,18
quaderno quadretti 200 pag	11c	7,20
evidenziatore giallo	23x	9,86

```
SELECT Articoli.descrizione, Percentuali.percentuale
FROM Articoli, Percentuali
WHERE Articoli.ID = Percentuali.id_articolo;
```

Il risultato sarà un elenco di record provenienti dalle due tabelle che vengono estratti soltanto se il campo ID ha un corrispettivo uguale nell'altra tabella.

Descrizione	Percentuale
penna a sfera	10%
refill rosso	3,5%
matita grafite 0,5 HB	10%
mina HH 0,5	3,5%

Nella FROM clause si indicano le tabelle oggetto d'interrogazione. Qualora esse siano più di una, devono essere separate dalla virgola.
Nella WHERE clause si indicano invece le condizioni per l'ottenimento delle tuple risultato. Anche in questa sezione, come nella target list, deve essere utilizzata la dot notation per indicare i campi che provengono da tabelle diverse. Se si interroga una sola tabella non è necessario utilizzare tale notazione, anche se i puristi di questo linguaggio tendono a farlo.

Gli operatori di confronto delle query

Espressioni condizionali
Le espressioni condizionali sono l'insieme di costanti, campi e operatori di confronto; opportunamente combinati tra loro, si definiscono espressioni condizionali in quanto il loro risultato può essere uno dei valori true/false.

Nelle query, se sono presenti **espressioni condizionali**, vengono usati gli operatori di confronto.

Una condizione è formata da tre parti:

campo | espressione	op. di confronto	campo | espressione | costante

Riportiamo gli operatori di confronto che possono essere utilizzati in una condizione nella tabella che segue.

Operatore	Significato
=	Uguale
<>	Diverso
<	Minore
>	Maggiore
<=	Minore o uguale
>=	Maggiore o uguale
LIKE	Confronto su stringhe
BETWEEN	Compreso tra due valori

La query SQL seguente mostra per esempio un nome visualizzato solo se il campo numero possiede un valore maggiore della costante 35:

```
SELECT T.nome
FROM Tabella T
WHERE T.numero > 35;
```

Il prossimo esempio mostra invece una condizione in cui a sinistra dell'operatore viene usato un attributo, mentre a destra compare una costante alfanumerica: vogliamo in questo caso ottenere un elenco che riporti il nome e i dati anagrafici del sig. Rossi. Il comando e il risultato corrispondente sono riportati di seguito.

```
SELECT P.cognome, P.nome, P.indirizzo, P.N_telefono
FROM Pazienti P
WHERE P.cognome = "Rossi";
```

Gli operatori Between e Like

L'operatore Like è usato quando si devono esprimere riferimenti a insiemi di caratteri. Tali riferimenti si ottengono utilizzando uno dei caratteri * (asterisco) oppure ? (punto di domanda), detti wildcard o caratteri jolly. Il carattere * indica "qualunque carattere", mentre il carattere ? indica "qualunque carattere, uno per ciascun punto di domanda".

L'operatore Between verifica invece se l'espressione è maggiore o uguale al valore minimo e minore o uguale al valore massimo.
La sintassi dei due operatori è la seguente:

```
LIKE carattere*

espressione BETWEEN valore minimo AND valore massimo
```

Vediamo alcuni esempi di utilizzo dei due operatori appena descritti.
Per ottenere tutti i nominativi di persone con altezza compresa tra 160 e 170 cm possiamo utilizzare l'operatore Between nella clausola WHERE utilizzando il seguente codice:

```
SELECT P.nome, P.cognome, P.altezza
FROM Persone P
WHERE P.altezza Between 160 AND 170;
```

Per ottenere tutti i nominativi di persone con un cognome che inizia per "R" e per "G", possiamo utilizzare l'operatore Like nella clausola WHERE utilizzando il seguente codice:

```
SELECT P.nome, P.cognome
FROM Persone P
WHERE P.cognome Like R* OR P.cognome Like G*;
```

Come possiamo notare, le due condizioni sono state unite da un operatore logico OR, che consente di accettare solo una delle due condizioni affinché la condizione complessiva risulti verificata.

Gli operatori aritmetici

Gli operatori aritmetici utilizzati nelle query consentono di eseguire le operazioni aritmetiche e sono presenti all'interno delle espressioni aritmetiche costituite da opportune combinazioni di attributi, costanti, parentesi e dei corrispondenti operatori. Queste espressioni possono essere inserite in una query dopo il comando SELECT, in una condizione WHERE o HAVING, oppure dopo la parola ORDER BY.
Si possono sommare due o più campi per ottenere un nuovo campo, per esempio:

```
SELECT (primoCampo + secondoCampo) AS "somma parziale"
FROM tabella;
```

In questo modo, avremo una colonna di nome "somma parziale" in cui saranno contenute tutte le somme dei due campi.

Per effettuare le altre operazioni (-, *, /, mod) dobbiamo agire in maniera analoga:

```
SELECT (primoCampo - secondoCampo) AS "sottrazione parziale"
FROM tabella;
SELECT (primoCampo * secondoCampo) AS "prodotto"
FROM tabella;
SELECT (primoCampo / secondoCampo) AS "risultato"
FROM tabella;
SELECT (primoCampo mod secondoCampo) AS "resto"
FROM tabella;
```

L'elevamento a potenza si ottiene elevando la base con l'esponente attraverso il carattere accento circonflesso (^):

```
SELECT (campo ^ esponente) AS "potenza";
```

La radice quadrata non negativa di un numero si ottiene con la funzione sqr():

```
SELECT sqr(campo);
```

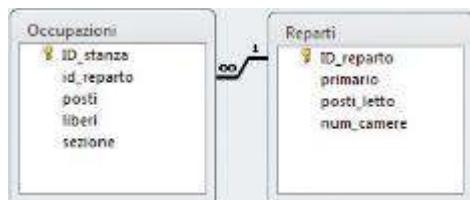
Gli operatori IN e IS NULL

L'operatore IN consente di abbreviare una condizione in cui lo stesso attributo viene uguagliato a un certo numero di costanti. Se, per esempio, con riferimento alle tabelle Pazienti e Visite del database PazientiVisite.mdb, volessimo un elenco che riporti il nome dei pazienti che sono in provincia di Torino, Milano e Como potremmo scrivere la query seguente:

```
SELECT P.nome, P.cognome
FROM Pazienti P, Visite V
WHERE P.ID_paziente = V.id_paziente
AND P.provincia IN ('CO','TO','MI');
```

L'operatore IS NULL consente di conoscere quali campi possiedono un valore nullo. Se vogliamo, per esempio, conoscere le visite mediche per le quali non è disponibile la pressione minima o massima, scriveremo la seguente query:

```
SELECT V.dataVisita, P.cognome
FROM Pazienti P, Visite V
WHERE P.ID_paziente = V.id_paziente AND
(V.pressioneMAX IS NULL OR V.pressioneMIN IS NULL);
```

Il prodotto cartesiano

Il prodotto cartesiano di due tabelle è la tabella risultante dalla congiunzione tra di esse che produce tutte le possibili combinazioni ordinate delle righe della prima tabella con tutte le righe della seconda. Tale risultato comporta un numero di righe pari alle righe (tuple) della prima

tabella, moltiplicate per le righe (tuple) della seconda, e un numero di colonne che è la somma delle colonne (attributi) della prima tabella più quelle della seconda.

ESEMPIO

STANZE OCCUPATE DI UN OSPEDALE

L'esempio seguente mostra il prodotto cartesiano tra le due tabelle Reparti e Occupazioni del database Ospedale.mdb:

```
SELECT O.ID_stanza, O.posti, O.liberi, O.sezione, R.ID_reparto, R.primario,
R.posti_letto
FROM Reparti AS R, Occupazioni AS O;
```

Eseguendo la query, otteniamo il prodotto cartesiano tra le due tabelle: la tabella Occupazioni contiene 10 record, mentre la tabella Reparti è di 7 righe, quindi il risultato è di 70 record (7 * 10).

prodotto

ID_stanza	posti	liberi	sezione	ID_repart	primario	posti_lett
1	4	0	m	1	verdi	8
1	4	0	m	2	rossi	8
1	4	0	m	3	manni	12
1	4	0	m	4	bianchi	8
1	4	0	m	5	gialli	10
1	4	0	m	6	colombo	22
1	4	0	m	7	esposito	30
2	4	1	f	1	verdi	8
2	4	1	f	2	rossi	8

Record: 1 di 70 · Nessun filtro · Cerca

VERIFICA... le competenze

AREA DIGITALE

Esercizi per l'approfondimento

ESERCIZI

1 Crea una tabella di nome Alunni con la struttura riprodotta a lato.
Il campo Maggiorenne contiene il valore 0 oppure 1, a seconda se l'allievo è maggiorenne o meno. Dopo aver popolato la tabella con alcuni record, crea le query SQL necessarie per eseguire i compiti di seguito indicati.

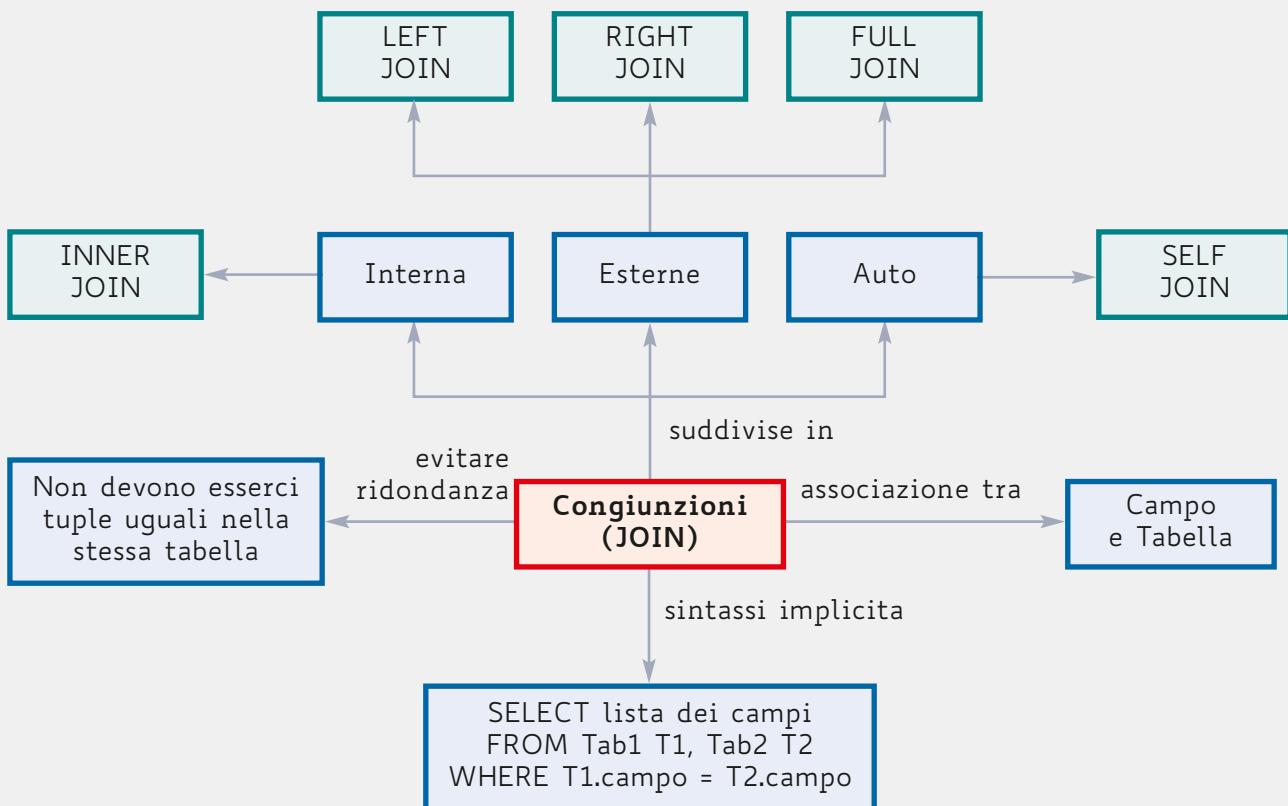
Alunni
- ID_Alunno
- Cognome
- Nome
- Indirizzo
- Classe
- MediaVoto
- Maggiorenne

- Elenco degli alunni maggiorenni delle varie classi.
- Elenco di tutti gli alunni della classe 4A.
- Aggiunta di 3 alunni alla classe 2B usando il comando SQL opportuno.
- Elenco di tutti gli studenti che hanno la media dei voti compresa tra 6 e 7 ma che non appartengono alla classe 5A e 5B.
- Elenco di tutti gli alunni con media tra 5 e 6 e tra 8 e 9.
- Elenco di tutti gli alunni di tutte le classi quarte e quinte (utilizza l'operatore Like).
- Elenco di tutti gli alunni con cognome che inizia per "L" e per "S".

2 Apri il database PazientiVisite.mdb.
- Crea una query che mostri l'elenco dei pazienti che non hanno codice ASL.
- Crea una query che mostri l'elenco delle visite: data visita, cognome del paziente visitato e peso misurato in occasione della visita.
- Crea una query che mostri l'elenco dei pazienti con differenza tra pressione minima e massima inferiore a 40.
- Crea una query che mostri l'elenco dei pazienti che pesano tra i 60 e i 70 kg.

3 Apri il database ArticoliFatture.mdb. Crea le query necessarie per eseguire i compiti di seguito indicati.
- Elenco di tutti gli articoli venduti nella fattura nr. 2.
- Elenco di tutte le fatture del cliente "rossi".
- Elenco degli articoli di clienti che hanno più di una fattura in date diverse.
- Elenco del numero e della data delle fatture per ciascun prodotto.
- Elenco delle fatture con più di 10 prodotti.
- Elenco degli articoli venduti ai clienti "rossi" e "gialli".

Confronta la tua soluzione con quella contenuta nel file ArticoliFatture_Solux.

4 Apri il database Biciclette.mdb. Crea le query necessarie per eseguire i compiti di seguito indicati.
- Elenco di tutte le bici prodotte da "Bianchi" e "Colnago", di tipo da "corsa" e "tandem".
- Elenco di tutti i produttori che hanno prodotto almeno una bicicletta di tipo "rampichino" e "sport".
- Elenco di tutte le biciclette prodotte in numero compreso tra 100 e 200 dai produttori "Gios" e "Olmo".
- Elenco di tutti i produttori che hanno prodotto biciclette con codice che inizia per "h" e "e".

5 Apri il database Film.mdb. Crea le query necessarie per eseguire i compiti di seguito indicati.
- Elenco di tutti i film con titolo che inizia per "s" e "u".
- Elenco di tutti i film usciti tra il 1996 e il 2001, che hanno incassato tra 50 e 100.
- Elenco di tutti i film in perdita economica (differenza tra costo e ricavo negativa).
- Elenco di tutti i film con rapporto tra costo e ricavo maggiore di 1.
- Elenco di tutti i fotografi che hanno realizzato film con incassi superiori al costo.
- Elenco di tutti i registi che hanno realizzato film prima dell'anno 2000.
- Elenco di tutti gli sceneggiatori che hanno partecipato alla realizzazione di film con costi compresi tra 200 e 300.

Le congiunzioni

IN QUESTA LEZIONE IMPAREREMO...

- a definire correlazioni tra tabelle secondo campi comuni
- a distinguere tra i vari tipi di congiunzione ammessi dal linguaggio SQL
- ad applicare congiunzioni multiple

didattica inclusiva

LEFT JOIN
RIGHT JOIN
FULL JOIN
INNER JOIN
Interna
Esterne
Auto
SELF JOIN
suddivise in
evitare ridondanza
Non devono esserci tuple uguali nella stessa tabella
Congiunzioni (JOIN)
associazione tra
Campo e Tabella
sintassi implicita

```
SELECT lista dei campi
FROM Tab1 T1, Tab2 T2
WHERE T1.campo = T2.campo
```

Le congiunzioni

Una regola fondamentale del modello relazionale, che elimina il problema della ridondanza, è quella di non consentire tuple uguali nelle stesse tabelle: il rispetto delle regole di normalizzazione rende possibile una gestione accurata delle tabelle e delle relazioni tra esse. Se per esempio volessimo memorizzare tutti i dati delle fatture emesse da un'azienda nella stessa tabella otterremmo un'inutile ripetizione dei dati anagrafici dei clienti, oppure degli articoli venduti. Per ottimizzare la struttura, usiamo tabelle diverse con informazioni accessibili mediante chiavi, pertanto possiamo affermare che una base di dati relazionale comporta in genere la presenza di molte tabelle.

La congiunzione (JOIN) è l'associazione tra un campo in una tabella e un campo dello stesso tipo di dati in un'altra tabella e viene utilizzata per collegare tra loro informazioni distribuite nelle varie tabelle. Le operazioni di selezione che consentono di ricavare da una relazione una tabella di risultati contenente solo le tuple che soddisfano una certa condizione vengono realizzate nel linguaggio SQL attraverso il costrutto SELECT.
La modalità più semplice per realizzare una JOIN tra due tabelle è descritta dalla sintassi seguente, che utilizza una sintassi implicita:

```
SELECT lista dei campi
FROM Tabella1 T1, Tabella2 T2
WHERE T1.campo = T2.campo;
```

ESEMPIO

GESTIONE UTENTI

Consideriamo due tabelle del database UtentiZone.mdb contenenti i dati anagrafici di alcune persone (Utenti) e le relative zone dove risiedono (Zone). Per visualizzare tutti gli utenti e le relative zone dove risiedono è necessario applicare una congiunzione tra le due tabelle. Bisogna cioè selezionare tutti gli utenti che hanno il campo Utenti.id_zona uguale al campo Zone.ID_zona:

```
SELECT U.cognome, U.nome, Z.descrizione
FROM Utenti U, Zone Z
WHERE U.id_zona = Z.ID_zona;
```

Si ottiene il risultato sotto riportato, dove per ciascun utente viene mostrata la relativa zona:

cognome	nome	descrizione
Rossi	Marco	Nord
Corti	Erika	Nord
Marsan	Marisa	Nord
Esposito	Raffaele	Sud
Capuano	Roberto	Sud
Verdini	Manuele	Centro
Russo	Marta	Nord-Est
Capuano	Alberto	Nord-Ovest
Capittu	Gavino	Isole

Se vi fossero degli utenti senza zona, cioè con una id_zona non presente nella tabella Zone, non apparirebbero nell'elenco.

Le congiunzioni esterne

Le congiunzioni esterne (OUTER JOIN) sono le congiunzioni in cui tutti i record di una tabella vengono aggiunti alle tuple risultato anche se non ci sono valori corrispondenti nel campo della seconda tabella con cui è stata creata la JOIN. I record della seconda tabella vengono combinati con quelli della prima tabella soltanto quando ci sono valori corrispondenti tra i campi in cui è stato creata la congiunzione.

Le congiunzioni esterne si suddividono in:

– LEFT JOIN (congiunzione sinistra), che estende tutte le tuple al primo operando, ovvero alla tabella a sinistra della congiunzione;
– RIGHT JOIN (congiunzione destra), che estende tutte le tuple al secondo operando, ovvero alla tabella a destra della congiunzione;
– FULL JOIN (congiunzione completa), che estende tutte le tuple a entrambi gli operandi ed è analoga a un prodotto cartesiano delle tabelle coinvolte, tuttavia non è implementato in Access.

LEFT JOIN

La sintassi della congiunzione sinistra (LEFT JOIN) è la seguente:

```
SELECT lista dei campi
FROM Tabella1 T1 LEFT JOIN Tabella2 T2
ON T1.attributo = T2.attributo;
```

La LEFT JOIN è la congiunzione in cui tutti i record del lato sinistro tra le due tabelle coinvolte nell'istruzione SQL vengono aggiunti alle tuple risultato, anche se non ci sono valori corrispondenti nel campo con cui è stata creata la JOIN nella tabella di destra. I record della tabella di destra vengono combinati con quelli della tabella di sinistra solo quando nei campi tra cui è stata creata la JOIN ci sono valori corrispondenti. Quando un record di sinistra non ha alcun record corrispondente, sul lato destro viene completata la riga con valori Null.

ESEMPIO

LEFT JOIN

Un tipico utilizzo di questa congiunzione è quello descritto di seguito.
Utilizziamo le due tabelle Persone e Recapiti del database PersoneRecapiti.mdb: la prima contiene un elenco di nominativi, mentre la seconda contiene i recapiti telefonici e ipertestuali.

Persone

ID_persona	cognome	nome	dataNascita
1	Rovi	Ortensia	10-apr-68
2	Margheritis	Rosa	13-mag-80
3	Colombo	Francesco	21-nov-54
4	Bianchi	Ciro	18-gen-60
5	Marelli	Antonio	10-mar-95

Tabella Persone

Recapiti

id_persona	n_telefon	fax	email	sito
1	02456789	02345123	ortensia.rovi@belvedere.com	
1	025612345	021234567		
2	0267890123			
2	055678890		rosa.margh@gmail.com	www.rosa.margeritis.it
5	345678964		an.marelli@tin.it	

Tabella Recapiti

Il codice che segue mostra come visualizzare i nominativi e i relativi recapiti:

```
SELECT P.nome, P.cognome, R.n_telefono, R.fax
FROM Persone P, Recapiti R
WHERE P.ID_persona = R.id_persona;
```

nome	cognome	n_telefono	FAX
Ortensia	Rovi	02456789	02345123
Ortensia	Rovi	025612345	021234567
Rosa	Margheritis	0267890123	
Rosa	Margheritis	055678890	
Antonio	Marelli	345678964	

Come possiamo notare, nella tabella di sinistra (Persone) vengono visualizzati solo i nominativi collegati alla tabella di destra (Recapiti), quindi vengono mostrati solo i nominativi che possiedono almeno un recapito: i record della tabella Persone che non possiedono almeno un corrispettivo nella tabella dei Recapiti non vengono mostrati.

Nel codice seguente, invece, dove viene utilizzata la LEFT JOIN, sono presenti tutti i record della tabella Persone, anche se non in possesso di alcun recapito.

```
SELECT P.nome, P.cognome, R.n_telefono, R.fax
FROM Persone P LEFT JOIN Recapiti R
ON P.ID_persona=R.id_persona;
```

nome	cognome	n_telefoni	fax
Ortensia	Rovi	02456789	02345123
Ortensia	Rovi	025612345	021234567
Rosa	Margheritis	0267890123	
Rosa	Margheritis	055678890	
Francesco	Colombo		
Ciro	Bianchi		
Antonio	Marelli	345678964	

Il risultato ottenuto è dunque fondamentalmente diverso, dal momento che l'elenco mostra anche i nominativi che non hanno alcun recapito.

RIGHT JOIN

La congiunzione destra (RIGHT JOIN) è la congiunzione in cui tutti i record del lato destro dell'istruzione SQL della query sono aggiunti alle tuple risultato, anche se non ci sono valori corrispondenti nel campo della tabella sinistra con cui è stata creata la JOIN. I record della tabella sinistra sono combinati invece con quelli della tabella destra solo quando ci sono valori corrispondenti nei campi tra cui è stata creata la JOIN. La sintassi è la seguente:

```
SELECT lista dei campi
FROM Tabella1 T1 RIGHT JOIN Tabella2 T2
ON T1.attributo = T2.attributo;
```

La congiunzione interna

La congiunzione interna (INNER JOIN) è la congiunzione in cui si combinano in un campo comune i record di due tabelle ogni volta che tali record contengono valori corrispondenti. Consente di ottenere una serie di tuple solo dalle tuple rispondenti a una condizione ON specificata esplicitamente. La sintassi di questa congiunzione è la seguente:

```
SELECT lista dei campi
FROM Tabella1 T1 INNER JOIN Tabella2 T2
ON T1.attributo { = | >| < | <> } T2.attributo;
```

ESEMPIO

CONGIUNZIONE INTERNA

Consideriamo le due tabelle Articoli e Sconti del database ArticoliSconti.mdb. Articoli contiene un elenco di prodotti con la descrizione e il prezzo, oltre a una chiave esterna (id_sconto) che rappresenta il codice della percentuale di sconto da applicare, mentre la tabella Sconti contiene per ogni codice il relativo sconto in percentuale. La query che segue mostra un elenco di tutti gli articoli scontati presenti nella tabella Articoli. Per ottenere questa query applichiamo la congiunzione interna:

```
SELECT A.descrizione, A.prezzo, S.sconto
FROM Articoli A INNER JOIN Sconti S
ON A.id_sconto = S.ID_sconto;
```

Auto-congiunzione

L'auto-congiunzione (SELF JOIN) è la congiunzione tra le colonne di una stessa tabella, in cui i record vengono combinati e aggiunti alle tuple risultato. Mediante questo particolare tipo di congiunzione possiamo creare un vincolo con la stessa tabella.

ESEMPIO

AUTO-CONGIUNZIONE

Supponiamo di avere a disposizione la tabella Impiegati del database Impiegati.mdb, contenente i nomi degli impiegati di un'azienda. Per conoscere chi è il superiore di un certo impiegato si utilizza un codice di identificazione (id_superiore). Il campo id_superiore è riferito al campo ID_impiegato della stessa tabella, pertanto in tal caso si parla di auto-congiunzione, cioè una congiunzione a elementi della stessa tabella. Se il codice vale 0 significa che non ci sono superiori per quel dipendente.

Impiegati

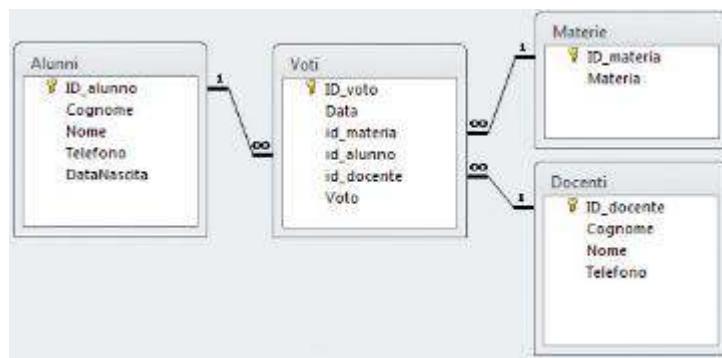
ID_impiegat	cognome	nome	id_superiori
1	Marchionne	Giovanni	0
2	Lupi	Pasquale	1
3	Sandrone	Pippo	1
4	Arlacchi	Samuele	1
5	Lupone	Gino	0
6	Soffice	Pamela	5
7	Colotti	Marisa	5
8	Marelli	Luca	5
9	Simoni	Algise	5
10	Carli	Elena	1

Per conoscere i dati dei dipendenti, con a fianco il relativo superiore, è necessario effettuare una query in cui introdurre una congiunzione alla stessa tabella attraverso uno "stratagemma": utilizzare due alias (I1 e I2) per la stessa tabella (Impiegati).

```
SELECT I1.nome, I1.cognome, I2.cognome AS "Superiore"
FROM Impiegati I1, Impiegati I2
WHERE I1.ID_superiore = I2.ID_impiegato;
```

Si ottiene il seguente risultato:

Query1

nome	cognome	superiore
Pasquale	Lupi	Marchionne
Pippo	Sandrone	Marchionne
Samuele	Arlacchi	Marchionne
Elena	Carli	Marchionne
Pamela	Soffice	Lupone
Marisa	Colotti	Lupone
Luca	Marelli	Lupone
Algise	Simoni	Lupone

Le congiunzioni multiple

Le operazioni di congiunzione sono binarie, ossia si applicano a coppie di tabelle. Infatti, negli esempi esaminati sinora, sono state prese in considerazione solo due tabelle alla volta. Le congiunzioni multiple sono invece quelle congiunzioni che coinvolgono un numero di tabelle maggiore di due. Per ottenere questo risultato è sufficiente eseguire le JOIN in maniera ordinata e coerente, a coppie di due. Prima si congiungono due tabelle, ottenendo così una nuova relazione temporanea, quindi il risultato di tale elaborazione viene congiunto con la terza tabella implicata nella relazione. Il meccanismo, ovviamente, può essere reiterato un numero indefinito di volte, ottenendo così delle congiunzioni tra quante tabelle si vuole.

ESEMPIO

ALUNNI E MATERIE

Consideriamo una struttura che registra i voti ottenuti dai vari alunni di una scuola (database AlunniDocentiMaterieVoti.mdb). Nella tabella Alunni vi sono i dati anagrafici relativi agli alunni, identificati da una chiave primaria (ID_alunno). La tabella Docenti possiede per ciascun docente una chiave primaria di identificazione (ID_docente). La tabella Materie contiene le informazioni circa le varie materie insegnate, con chiave primaria rappresentata dal campo ID_materia. La tabella Voti, infine, contiene 3 chiavi esterne per collegare il voto alle tabelle Docenti, Alunni e Materie. Di seguito è riportata la struttura delle quattro tabelle.

La congiunzione multipla viene utilizzata in tutti i casi nei quali è necessario associare tra loro più di due tabelle e avviene mediante dei passaggi distinti. Per conoscere le valutazioni ricevute dagli alunni nelle diverse materie dobbiamo realizzare una congiunzione multipla tra le tabelle Alunni, Voti e Materie. Per realizzare una congiunzione multipla è necessario racchiudere tra parentesi la congiunzione più interna: nel nostro caso, metteremo tra parentesi la congiunzione tra Alunni e Materie.

La query riportata a pagina seguente può essere letta così: una prima congiunzione restituisce alcune tuple risultato, ed è posta tra parentesi; quindi, sulle tuple ottenute, viene effettuata una seconda congiunzione che dà luogo alle successive tuple.

```
SELECT A.cognome, A.nome, V.voto, M.materia
FROM
Alunni A LEFT JOIN Voti V ON
A.ID_alunno=V.id_alunno
(   LEFT JOIN Materie M ON
   V.id_materia=M.ID_materia);
```

cognome	nome	voto	materia
Beretta	Marco	6	Inglese
Beretta	Marco	5	Tedesco
Capozzi	Marina	7	Inglese
Capozzi	Marina	7	Tedesco
Perti	Monica	5	Inglese
Perti	Monica	4	Tedesco

In questo esempio abbiamo utilizzato la congiunzione sinistra per fare in modo che potesse apparire anche un allievo senza valutazioni, senza cioè alcun record correlato nella tabella Voti.

Passiamo ora alla realizzazione della query che consente anche di visualizzare il nome del docente che ha assegnato quei voti. Applicando i criteri appena visti, aprendo cioè un'altra parentesi, possiamo associare le tuple ottenute anche a una quarta tabella: Docenti.

```
SELECT A.cognome, A.nome, V.voto, M.materia, D.cognome AS [cognome Prof],
D.nome AS [nome Prof]
FROM ((Alunni A LEFT JOIN Voti V ON A.ID_alunno=V.id_alunno)
   LEFT JOIN Materie M ON V.id_materia=M.ID_materia)
      LEFT JOIN Docenti D ON V.id_docente=D.ID_docente;
```

Abbiamo aggiunto un alias al posto del cognome e nome del docente scrivendolo tra parentesi quadre precedute dal prefisso AS ([cognomeProf] e [nomeProf]).
Come possiamo notare, appare un elenco nel quale vi sono i campi delle diverse tabelle coinvolte:

cognome	nome	voto	materia	cognome Prof	nome Prof
Beretta	Marco	6	Inglese	Russo	Samuela
Beretta	Marco	5	Tedesco	Bianchi	Maria
Capozzi	Marina	7	Inglese	Russo	Samuela
Capozzi	Marina	7	Tedesco	Bianchi	Maria
Perti	Monica	5	Inglese	Russo	Samuela
Perti	Monica	4	Tedesco	Bianchi	Maria

In pratica, valgono le stesse regole usate per le espressioni: prima vengono eseguite le associazioni poste nelle parentesi più interne, e poi via via fino alle più esterne, secondo uno schema che può essere generalizzato come segue:

```
SELECT elenco campi
FROM
   (...( Tabella1 T1 JOIN Tabella2 T2 ON T1.attributo=T2.attributo)
      JOIN Tabella3 T3 ON ...=T3.attributo)
         ...
            JOIN Tabellan Tn ON ...=Tn.attributo)
```

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 **Indica quale, tra le seguenti query, rappresenta una selezione + proiezione:**
- **a** `SELECT * FROM Colleghi;`
- **b** `SELECT cognome, nome FROM Colleghi;`
- **c** `SELECT cognome, nome FROM Colleghi WHERE città="Como";`
- **d** `SELECT C.nome FROM Colleghi C, Filiale F WHERE C.ID_filiale=F.ID;`

2 **Indica quale, tra le seguenti query, rappresenta una proiezione:**
- **a** `SELECT * FROM Colleghi;`
- **b** `SELECT cognome, nome FROM Colleghi;`
- **c** `SELECT cognome, nome FROM Colleghi WHERE città="Como";`
- **d** `SELECT C.nome FROM Colleghi C, Filiale F WHERE C.ID_filiale=F.ID;`

3 **Indica quale tra le seguenti query rappresenta una congiunzione + proiezione:**
- **a** `SELECT * FROM Colleghi;`
- **b** `SELECT cognome, nome FROM Colleghi;`
- **c** `SELECT cognome, nome FROM Colleghi WHERE città="Como";`
- **d** `SELECT C.nome FROM Colleghi C, Filiale F WHERE C.ID_filiale=F.ID;`

4 **Che tipo di congiunzione è espressa dalla seguente query?**

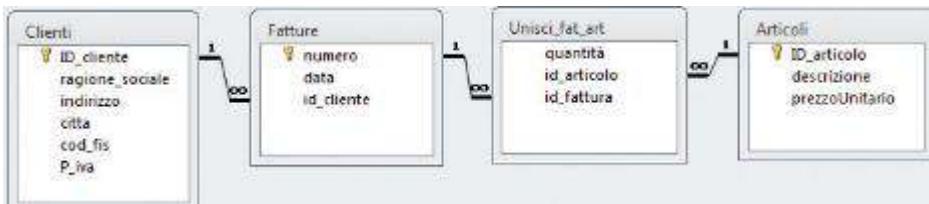
```
SELECT P.nome, P.cognome, P.pneumatico,
A.marca_vettura
FROM (Piloti P INNER JOIN Automobili A
ON P.id_vettura=A.ID_vettura)
   INNER JOIN Pneumatici P ON
   A.id_pneumatico=P.ID.pneumatico
```

- **a** Congiunzione errata
- **b** Congiunzione esterna
- **c** Congiunzione interna
- **d** Congiunzione multipla

5 **Indica quale tra le seguenti è una SELF JOIN corretta:**
- **a** `SELECT A1.cognome, A2.cognome FROM Alunni A1, Alunni A2 WHERE A1.id_rapp=A2.cognome;`
- **b** `SELECT A1.cognome FROM Alunni A1, Alunni A2, WHERE A1.id_rappr=A2.ID_alunno;`
- **c** `SELECT * FROM Alunni A1, Alunni A2 WHERE A1.id_rapp=A2.ID;`
- **d** `SELECT A2.cognome FROM Alunni A1, Alunni A2, WHERE A1.id_rappr=A2.ID_alunno;`

6 **Quale congiunzione tra le seguenti è interna?**
- **a** INNER JOIN
- **b** LEFT JOIN
- **c** RIGHT JOIN
- **d** SELF JOIN

VERIFICA... le competenze

AREA DIGITALE

ESERCIZI

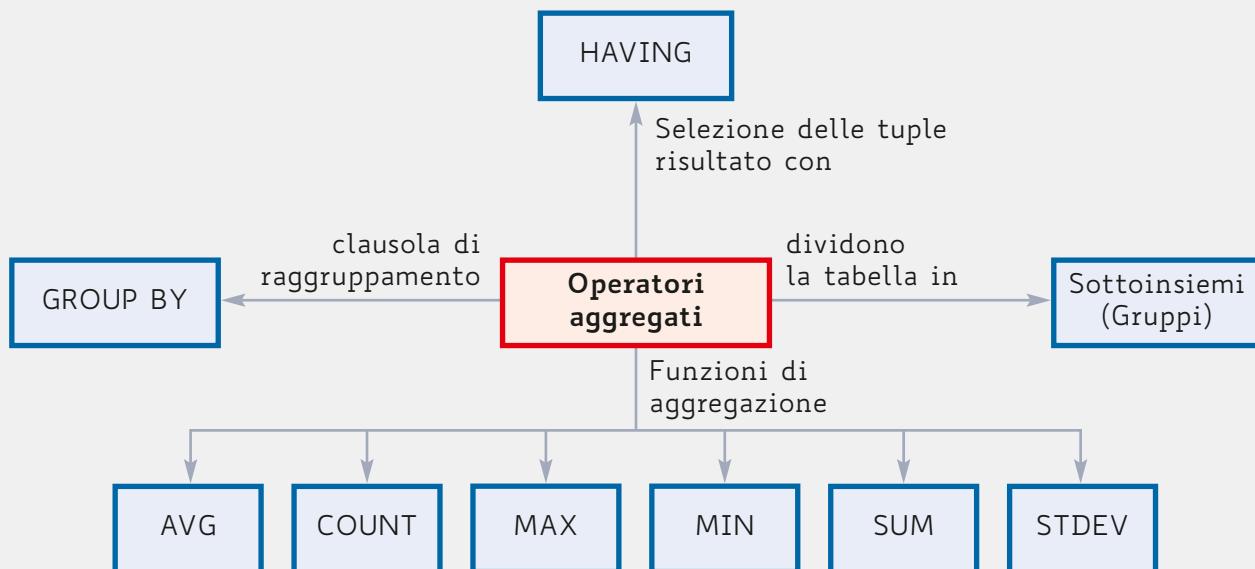
1 Apri il database Fatturazione.mdb, che possiede la seguente struttura:

Crea le query necessarie per eseguire i compiti di seguito descritti.
- Visualizzazione degli articoli fatturati nell'anno 2003.
- Visualizzazione di tutte le fatture emesse per i clienti di Como e Milano.
- Visualizzazione, per ogni articolo di prezzo unitario compreso tra 180 e 350 euro, del cliente e della data in cui è stato venduto, anche per prodotti mai venduti, usando quindi una congiunzione sinistra.
- Come la query precedente, ma per i soli articoli che sono stati effettivamente fatturati.

Confronta i tuoi risultati con quelli del file Fatturazione_corretta.

2 Crea una tabella Giocatori con la seguente struttura:
ID_giocatore
Nome
Cognome
Squadra
id_capitano

Crea quindi una query che visualizzi l'elenco dei giocatori con a fianco il nome del capitano usando un'auto congiunzione (i capitani possiedono il campo id_capitano = 0).

3 Definisci un database mediante codice SQL per memorizzare i dati dei calciatori dei campionati di serie A, B e C di un particolare anno. Crea le squadre di calcio, dove per ogni squadra vi sono più calciatori. Ogni giocatore ha un codice identificativo, così come ogni squadra. Per ogni squadra esiste poi un capitano, scelto tra i calciatori. Per ciascun calciatore indica il valore di mercato e l'ingaggio annuale. Ogni squadra ha un indirizzo, un budget annuale e un numero di abbonati. Ogni calciatore ha un procuratore e più calciatori possono avere lo stesso procuratore. Dopo aver creato le tabelle, definisci le seguenti query di congiunzione in SQL:
- elenco di tutti i calciatori che militano in squadre che hanno più di 10.000 abbonati;
- elenco di tutti i procuratori di giocatori di serie B e C;
- elenco di tutti i calciatori, anche quelli senza procuratore;
- come sopra, ma riferito ai soli giocatori di serie A;
- elenco di tutte le squadre con i relativi procuratori per ogni giocatore (evita ripetizioni dello stesso procuratore);
- elenco dei procuratori senza ripeterne il nome;
- elenco di tutti i procuratori disoccupati;
- elenco di tutti i calciatori disoccupati che valgono più di 10.000 euro.

Gli operatori aggregati

IN QUESTA LEZIONE IMPAREREMO...

- a utilizzare i principali operatori di aggregazione
- a distinguere tra l'utilizzo degli operatori aggregati con o senza raggruppamento
- ad applicare i raggruppamenti con le congiunzioni
- a rappresentare raggruppamenti con condizioni esterne

MAPPA CONCETTUALE

didattica inclusiva

HAVING
Selezione delle tuple risultato con
GROUP BY
clausola di raggruppamento
Operatori aggregati
dividono la tabella in
Sottoinsiemi (Gruppi)
Funzioni di aggregazione
AVG
COUNT
MAX
MIN
SUM
STDEV

Gli operatori aggregati

Le operazioni di aggregazione consentono di raggruppare le tuple per effettuare calcoli specifici quali, per esempio, sommatorie, oppure conteggi, o ancora semplici calcoli statistici. Possiamo dividere la tabella risultante da una query con operatori aggregati in sottoinsiemi, raggruppando le tuple contenenti gli stessi valori per un insieme di attributi.
Gli operatori di aggregazione, a differenza degli operatori matematici, che agiscono soltanto sulla tupla in corso di elaborazione, possono essere applicati a più tuple della tabella e si caratterizzano per il fatto di restituire un valore in corrispondenza di un gruppo di valori o dei valori che formano una colonna. Elenchiamo di seguito i principali operatori di aggregazione:
– AVG (nome campo), che calcola la media aritmetica;
– COUNT (espressione | *), che conta le righe;
– MAX (espressione), che calcola il valore massimo;
– MIN (espressione), che calcola il valore minimo;
– SUM (nome campo numerico), che calcola la somma totale;
– STDEV (nome campo numerico), che calcola la deviazione standard.

Gli operatori aggregati possono comparire solo dopo SELECT e HAVING.

ESEMPIO

CONTEGGIO DI PRODOTTI

Per calcolare quanti articoli sono presenti complessivamente nella tabella Prodotti del database Supermarket.mdb utilizziamo l'operatore di aggregazione COUNT.

```
SELECT COUNT(ID_prodotto) AS [Tot prodotti]
FROM Prodotti;
```

Si ottiene il risultato mostrato a lato.

L'operatore di aggregazione COUNT agisce sull'intera tabella e non solo sulla tupla in corso di elaborazione restituendo un unico risultato.

Gli operatori di aggregazione restituiscono una tabella, anche se in questo caso è rappresentata da una sola riga e una sola colonna.

A eccezione di COUNT, tutti gli altri operatori di aggregazione non considerano i valori nulli ai fini del calcolo.

ESEMPIO

CALCOLO TEMPO MEDIO DI ATTESA

Vogliamo calcolare il tempo medio di attesa per la consegna di una nuova auto: a tale scopo, usiamo una query simile alla precedente, sostituendo l'attributo ID_prodotto con GG_attesa.
Riportiamo qui sotto la tabella Automobili del database Automobili.mdb e la relativa query.

Automobili

ID_auto	Marca	Modello	Tipo	Prezzo	GG_attesa
1	Fiat	Punto		€ 10.000,00	12
2	Seat	Ibiza		€ 12.000,00	25
3	Audi	Q2		€ 29.000,00	45
4	Mercedes	A180		€ 19.000,00	30
5	Volvo	XC60		€ 38.000,00	70
6	BMW	X1	4X4	€ 44.500,00	56
7	Citroen	Picasso	Cabriolet	€ 28.000,00	40
8	Peugeot	408	Cabriolet	€ 32.000,00	50
9	Fiat	500x	4X4	€ 26.000,00	50
10	Jeep	Renegade	4X4	€ 33.000,00	80
11	Jeep	Compass	4X4	€ 53.000,00	90
12	Chrysler	Voyager		€ 67.000,00	110

```
SELECT AVG(GG_attesa) AS [Tempo medio di attesa]
FROM Automobili;
```

Il risultato della query, dato dal rapporto tra il totale dei giorni e il numero di valori che in quell'attributo sono diversi da NULL, è riportato qui a lato.

Regole di visibilità

I campi presenti nella target list di una query di aggregazione devono sottostare ad alcune regole, chiamate regole di visibilità.
Vediamo un esempio che ci permette di comprendere il funzionamento di tali regole: supponendo di voler calcolare nella tabella Automobili del precedente esempio il prezzo medio delle autovetture di tipo “4X4”, ci rendiamo facilmente conto che la seguente query è assolutamente errata:

```
SELECT AVG(Prezzo), Modello
FROM Automobili
WHERE Tipo = "4X4";
```

Il risultato è un messaggio di errore, diverso a seconda del tipo di DBMS utilizzato:

L’errore è dovuto alla presenza, nella target list, sia di un campo (Modello) che di un operatore di aggregazione (AVG), che fa sì che il calcolo comporti la perdita delle componenti della tabella originale in termini di righe. Volendo dare una risposta in termini più semplici, possiamo chiederci a chi appartiene la media: ovviamente non a un record in particolare, essendo la media di tutti i prezzi delle auto “4X4”.
Il problema, come vedremo più avanti, viene risolto ricorrendo alla clausola GROUP BY.

L’operatore COUNT

Esaminiamo ora più in dettaglio l’operatore COUNT. Esso, in pratica, riporta il conteggio totale delle tuple ottenute dalla query di selezione. Questo operatore può essere utilizzato su qualunque tipo di dato.
Sempre con riferimento alla tabella Automobili, eseguiamo la seguente query:

```
SELECT COUNT(Tipo)
FROM Automobili;
```

Il risultato che si ottiene è 6: anche se in realtà esistono più tipi uguali, viene infatti calcolato il totale di tutti i record che compongono la tabella e che non sono NULL.
Una caratteristica dell’operatore COUNT è l’argomento da indicare tra parentesi, che detiene un ruolo particolare in quanto il conteggio viene eseguito soltanto per le righe che non contengono valori NULL nell’argomento indicato tra parentesi.

Vediamo adesso come utilizzare l’operatore COUNT unitamente a condizioni WHERE, per conoscere, per esempio, quante sono le auto di marca “Jeep” e di tipo “cabriolet”:

```
SELECT COUNT(GG_attesa)
FROM Automobili
WHERE Marca="Jeep" AND Tipo="cabriolet";
```

Il parametro dell’operatore COUNT può essere l’asterisco (*), oppure un campo specifico. Se viene indicato un campo specifico deve essere stato dichiarato di tipo NOT NULL.

Gli operatori MAX e MIN

MAX e MIN restituiscono i valori più alti e più bassi contenuti all'interno di una colonna. Per i campi numerici restituisce il valore numerico, mentre per quelli testuali seleziona il campo che, in ordine alfabetico, è ultimo (max) o primo (min), secondo la tabella dei codici ASCII. È possibile, inoltre, usare come argomento degli operatori MIN e MAX anche espressioni matematiche. Sempre con riferimento alla tabella Automobili, scriviamo la seguente query per conoscere il prezzo più basso delle automobili in vendita:

```
SELECT MIN(Prezzo)
FROM Automobili;
```

Anche per gli operatori MAX e MIN valgono le regole viste per COUNT: per esempio, non è possibile inserire altri nomi di campo nella target list; infatti, la seguente query è errata:

```
SELECT marca, MAX(Prezzo)
FROM Automobili;
```

Nella target list devono comparire un operatore e un attributo: il significato logico di tale query è, pertanto, da definirsi assurdo.

È invece possibile associare più operatori aggregati; per esempio, se vogliamo calcolare il conteggio sui giorni di attesa e nel contempo il valore massimo, sempre riferito ai giorni di attesa, avremo:

```
SELECT MAX(GG_attesa), COUNT(GG_attesa)
FROM Automobili;
```

Si possono inoltre utilizzare gli alias anche per gli operatori di aggregazione, soprattutto per fornire una migliore leggibilità dei risultati.

ESEMPIO

STIPENDI MASSIMI E MINIMI

Apriamo il database Stipendi.mdb e consideriamo la tabella Stipendi, che contiene i seguenti record riferiti agli impiegati di una azienda:

ID_impiegat	cognome	nome	id_superior	Stipendio
1	Marchionne	Giovanni	0	€ 4.200,00
2	Lupi	Pasquale	1	€ 2.345,00
3	Sandrone	Pippo	1	€ 678,00
4	Arlacchi	Samuele	1	€ 1.800,00
5	Lupone	Gino	0	€ 2.700,00
6	Soffice	Pamela	5	€ 2.500,00
7	Colotti	Marisa	5	€ 1.200,00
8	Marelli	Luca	5	€ 2.500,00
9	Simoni	Algise	5	€ 1.200,00
10	Carli	Elena	1	€ 800,00

Vogliamo individuare gli stipendi più alti e più bassi tra tutti gli impiegati, oltre al numero complessivo di impiegati presenti. Scriviamo pertanto la seguente query:

```
SELECT MAX(Stipendio) AS [Massimo], MIN(Stipendio) AS [Minimo],
COUNT(Stipendio) AS [Impiegati totali]
FROM Stipendi;
```

Massimo	Minimo	Impiegati totali
€ 4.200,00	€ 678,00	10

La tupla risultato è riportata nell'immagine a lato.

Vogliamo adesso calcolare separatamente lo stipendio massimo dei responsabili e del resto dei dipendenti. Per ottenere questa informazione, scriviamo la seguente query a partire dalla tabella Stipendi, che viene "sdoppiata" secondo la tecnica della SELF JOIN.

```
SELECT MAX(S1.Stipendio) AS [Stipendio Max Rappr], MAX(S2.Stipendio) AS
[Stipendio Max Altri]
FROM Stipendi S1, Stipendi S2
WHERE S1.ID_Impiegato=S2.id_superiore;
```

Vediamo ora come applicare alcuni operatori aritmetici nella query. Per conoscere, per esempio, la differenza tra stipendio minimo e massimo degli impiegati, rapportata in percentuale, dobbiamo effettuare la divisione tra la differenza tra il valore massimo e minimo degli stipendi e lo stipendio massimo.

```
SELECT str((MAX(stipendio)-MIN(stipendio))/(MAX(stipendio))*100)+"%" AS
percentuale
FROM Stipendi;
```

percentuale
83.8571428571428%

La tupla risultato è riportata nell'immagine a lato.

L'operatore SUM

L'operatore SUM è rigorosamente applicabile solo a campi numerici e restituisce la somma dei valori contenuti nella colonna prescelta. La sua sintassi è assai semplice:

```
SELECT SUM(campo numerico)
FROM nome tabella
[WHERE condizione];
```

In base a quanto detto per gli altri operatori, è facilmente intuibile come utilizzarlo: valgono infatti tutte le regole indicate per MIN e MAX. Vediamo un esempio.

Sempre con riferimento alla tabella Automobili (database Automobili.mdb) scriviamo una query per individuare il totale dei prezzi di tutte le autovetture di tipo "4X4". La riportiamo di seguito, insieme al risultato ottenuto:

```
SELECT SUM(Prezzo) AS [somma dei prezzi]
FROM Automobili
WHERE tipo = "4X4";
```

somma dei prezzi
€ 156.500,00

Gli operatori AVG e STDEV

Questi operatori sono assai utili ai fini statistici e permettono il calcolo della media aritmetica (AVG) e dello scarto quadratico medio o deviazione standard (STDEV). Vediamone la sintassi:

```
SELECT AVG(campo numerico)
FROM nome tabella
[WHERE condizione];
```

Per esempio, vogliamo calcolare la media di tutti gli stipendi e lo scarto quadratico medio relativamente a tutti gli impiegati presenti nella tabella Stipendi del database Stipendi.mdb:

```
SELECT AVG(Stipendio) AS Media, STDEV(Stipendio) AS Devianza
FROM Stipendi;
```

Si ottiene la seguente tupla risultato:

Media	Devianza
€ 1.992,30	1077,86394215

GROUP BY
Determina la creazione di una singola tupla per tutti i record con il medesimo contenuto nell'attributo (o negli attributi) specificato dopo la parola chiave GROUP BY.

La clausola GROUP BY

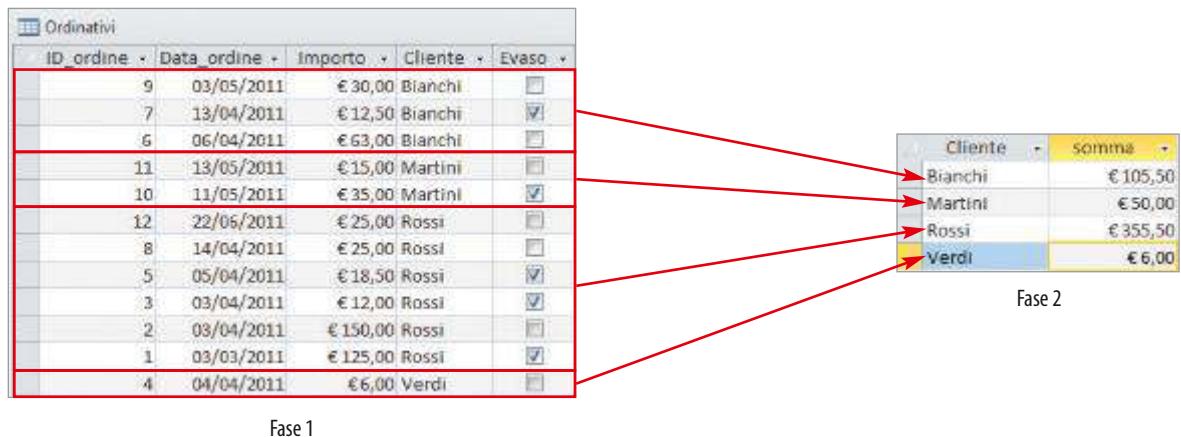
La clausola **GROUP BY** serve per raggruppare e per elaborare in modo uniforme diverse righe che nella tabella origine hanno valori uguali in una determinata colonna.

La sintassi della clausola GROUP BY è la seguente:

```
SELECT lista dei campi | espressioni
FROM tabelle
[WHERE condizione]
[GROUP BY lista campi di raggruppamento]
[HAVING condizioni aggregate]
[ORDER BY lista campi di ordinamento]
```

Mediante le considerazioni che seguono, facciamo un po' di chiarezza sulle priorità durante l'esecuzione di una selezione con raggruppamento.

1 Prima di tutto avviene l'esecuzione della JOIN, se essa è stata inserita all'interno del codice; altrimenti si passa al punto 2.
2 Avviene la selezione sulle tuple delle tabelle originali con la condizione opzionale presente dopo la clausola WHERE.
3 Eventuale raggruppamento in base ai campi specificati dopo la clausola GROUP BY.
4 Lista dei valori specificati dopo la SELECT (creazione della target list).
5 Eventuale selezione delle tuple che soddisfano la condizione aggregata presente dopo la clausola HAVING.

Per introdurre il concetto di raggruppamento prendiamo in esame la tabella Ordinativi presente nel database Ordinativi.mdb, contenente i record indicati:

Ordinativi

ID_ordine	Data_ordine	Importo	Cliente	Evaso
1	03/03/2011	€ 125,00	Rossi	☑
2	03/04/2011	€ 150,00	Rossi	☐
3	03/04/2011	€ 12,00	Rossi	☑
4	04/04/2011	€ 6,00	Verdi	☐
5	05/04/2011	€ 18,50	Rossi	☑
6	06/04/2011	€ 63,00	Bianchi	☐
7	13/04/2011	€ 12,50	Bianchi	☑
8	14/04/2011	€ 25,00	Rossi	☐
9	03/05/2011	€ 30,00	Bianchi	☐
10	11/05/2011	€ 35,00	Martini	☑
11	13/05/2011	€ 15,00	Martini	☐
12	22/06/2011	€ 25,00	Rossi	☐

Vediamo cosa accade eseguendo la query seguente, che ha lo scopo di raggruppare tutti i clienti sommando poi gli importi di tutti gli ordini effettuati da ciascuno di essi.

```
SELECT Cliente, SUM(Importo) AS somma
FROM Ordinativi
GROUP BY Cliente;
```

Cliente	somma
Bianchi	€ 105,50
Martini	€ 50,00
Rossi	€ 355,50
Verdi	€ 6,00

Si verificano due passaggi; nel primo le tuple vengono ordinate rispetto al cliente, poi, per ogni cambiamento di cliente, secondo un metodo detto a rottura di codice, viene effettuata la somma di tutti gli importi, con il seguente risultato:

Ordinativi

ID_ordine	Data_ordine	Importo	Cliente	Evaso
9	03/05/2011	€ 30,00	Bianchi	☐
7	13/04/2011	€ 12,50	Bianchi	☑
6	06/04/2011	€ 63,00	Bianchi	☐
11	13/05/2011	€ 15,00	Martini	☐
10	11/05/2011	€ 35,00	Martini	☑
12	22/06/2011	€ 25,00	Rossi	☐
8	14/04/2011	€ 25,00	Rossi	☐
5	05/04/2011	€ 18,50	Rossi	☑
3	03/04/2011	€ 12,00	Rossi	☑
2	03/04/2011	€ 150,00	Rossi	☐
1	03/03/2011	€ 125,00	Rossi	☑
4	04/04/2011	€ 6,00	Verdi	☐

Fase 1

Cliente	somma
Bianchi	€ 105,50
Martini	€ 50,00
Rossi	€ 355,50
Verdi	€ 6,00

Fase 2

Analizziamo ora i passaggi di una query di raggruppamento contenente una condizione WHERE. Per esempio, sempre con riferimento alla tabella Ordinativi, si vogliono visualizzare i totali degli ordini evasi, suddivisi per cliente. Innanzitutto riportiamo il codice della query:

```
SELECT Cliente, SUM(Importo) AS somma
FROM Ordinativi
WHERE Evaso
GROUP BY cliente;
```

Cliente	importo	evaso
Rossi	€ 125,00	☑
Rossi	€ 12,00	☑
Rossi	€ 18,50	☑
Bianchi	€ 12,50	☑
Martini	€ 35,00	☑

Questa query viene eseguita in tre fasi: prima viene valutata la condizione WHERE, ottenendo quindi una serie intermedia di tuple.

Su questa serie di tuple avviene il raggruppamento in base al cliente, formando così la seconda serie intermedia e quindi la sommatoria che fornisce i risultati definitivi.

Cliente	somma
Bianchi	€ 12,50
Martini	€ 35,00
Rossi	€ 155,50

La visibilità dei campi nella clausola GROUP BY

È importante sottolineare che la visibilità dei campi oggetto di raggruppamento è assai importante nella comprensione di questo argomento. In ciascuna delle tuple ottenute dal raggruppamento, analogamente a quanto avviene quando si utilizzano gli operatori aggregati, non sono infatti più disponibili i dati delle tuple originali. Al loro posto, possono essere elencati i seguenti valori:
– tutti gli operatori di aggregazione;
– qualunque attributo specificato di seguito alla clausola GROUP BY.

Sempre con riferimento alla tabella Ordinativi, la seguente query è pertanto da considerarsi non corretta, in quanto l'attributo di raggruppamento (Cliente) non compare nella target list e ciò non permette di raggruppare la tabella in quanto manca fisicamente il campo:

```
SELECT Importo
FROM Ordinativi
GROUP BY Cliente;
```

Anche nel seguente caso la query è da considerarsi non corretta, dal momento che nella target list compare il campo Indirizzo della tabella Clienti, che tuttavia non appare tra gli attributi di raggruppamento dopo il GROUP BY:

```
SELECT O.Cliente, COUNT(*), C.Indirizzo
FROM Ordinativi O INNER JOIN Clienti C ON (O.Cliente = C.Cliente)
GROUP BY O.Cliente;
```

La seguente query è invece assolutamente corretta, in quanto nella target list ci sono attributi presenti anche nella clausola GROUP BY:

```
SELECT O.Cliente, COUNT(*), C.Indirizzo
FROM Ordinativi O INNER JOIN Clienti C ON (O.Cliente = C.Cliente)
GROUP BY O.Cliente, C.Indirizzo;
```

HAVING
La clausola HAVING permette di porre condizioni finali sulla serie di tuple risultato fornite da una query contenente tuple accorpate.

Le condizioni HAVING

Spesso è necessario esprimere alcune condizioni sulle tuple risultato. In questo caso, oltre alla clausola GROUP BY, la condizione deve essere espressa obbligatoriamente ricorrendo alla clausola **HAVING**, e non alla clausola WHERE.

La sintassi di tale clausola è già stata vista in precedenza (*vedi* sintassi GROUP BY pag. 228); ricordiamo soltanto che va sempre messa alla fine di una query di raggruppamento.

```
SELECT elenco campi
...
GROUP BY
HAVING condizione
```

ESEMPIO

FILTRARE I RISULTATI DI UNA QUERY CON HAVING
Dalla tabella Ordinativi vogliamo estrarre mediante query i clienti che hanno, in totale, effettuato acquisti superiori a 100 €. Per comprendere meglio come agisce la clausola HAVING, scomponiamo la query in due sequenze di operazioni: nella prima effettuiamo una query di raggruppamento per nome Cliente, dove calcolare, per ciascun gruppo ottenuto, la somma degli Importi:

```
SELECT Cliente, SUM(Importo) AS [Totale importi]
FROM Ordinativi
GROUP BY Cliente;
```

Cliente	Totale impo
Bianchi	€ 105,50
Martini	€ 50,00
Rossi	€ 355,50
Verdi	€ 6,00

Si ottengono le tuple risultato riportate a lato.

Quindi, sulle tuple risultato ottenute, effettuiamo l'ulteriore "filtraggio" con la clausola HAVING:

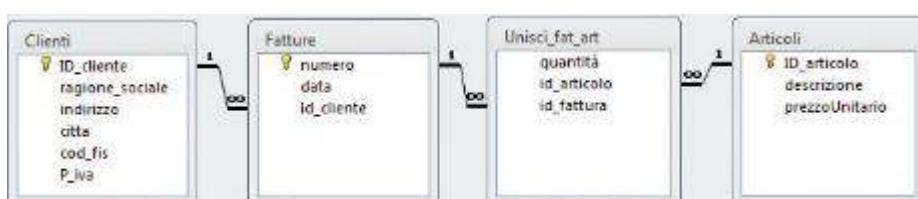
```
SELECT Cliente, SUM(Importo) AS [Totale importi]
FROM Ordinativi
GROUP BY Cliente
HAVING SUM(Importo)>100;
```

Cliente	Totale impo
Bianchi	€ 105,50
Rossi	€ 355,50

Come possiamo notare nell'immagine a lato, appariranno solo le tuple che soddisfano la condizione SUM(Importo)>100.

La clausola HAVING permette dunque di considerare solo le tuple che soddisfano la condizione relativamente alle righe ottenute dopo il raggruppamento e l'eventuale calcolo.

Vediamo adesso come detta clausola si comporta quando le condizioni sono multiple, per esempio se si vogliono conoscere, tra tutti i clienti che hanno eseguito più di due ordini, quelli con una media di importi minore di 150:

```
SELECT COUNT(Importo)AS [Totale importi], Cliente, AVG(Importo) AS media
FROM Ordinativi
GROUP BY Cliente
HAVING AVG(Importo)<150 and COUNT(Importo)>2;
```

Totale impo	Cliente	media
3	Bianchi	€ 35,17
6	Rossi	€ 59,25

Come possiamo notare nell'immagine a lato, si ottengono solo due clienti che soddisfano entrambe le condizioni.

ESEMPIO

RAGGRUPPAMENTO CON CONGIUNZIONI MULTIPLE

Realizziamo query di aggregazione con congiunzioni multiple, prendendo in esame le seguenti tabelle presenti nel database Fatturazione.mdb:

Per effettuare il raggruppamento con congiunzioni multiple è necessario aggiungere gli operatori di raggruppamento alle tuple ottenute dalla congiunzione.

Per esempio, vogliamo calcolare quanti articoli sono stati venduti a un determinato cliente. Per fare questo, occorre innanzitutto effettuare una selezione e proiezione con condizione e congiunzione delle tabelle; quindi, sulle tuple ottenute, effettuarne la somma mediante l'operatore SUM. Il codice e il risultato della query sono riportati a pagina seguente.

```
SELECT SUM(U.quantità) AS [Totale articoli Nobis]
FROM (Clienti C INNER JOIN Fatture F ON C.ID_cliente=F.id_cliente)
   INNER JOIN Unisci_fat_art U ON U.id_fattura=F.numero
      WHERE C.ragione_sociale="nobis";
```

Totale articoli Nobis
27

Come possiamo notare, non è stato utilizzata la clausola GROUP BY, in quanto la condizione WHERE ha operato una selezione delle tuple con ragione_sociale uguale a "nobis" prima del raggruppamento.

Vediamo adesso di calcolare il totale di merci vendute per ciascun cliente: in questo caso, risulta obbligatorio l'uso di GROUP BY per raggruppare per cliente. Il codice e il risultato della query sono riportati di seguito:

```
SELECT C.ragione_sociale, SUM(U.quantità) AS [Totale articoli]
FROM (Clienti C INNER JOIN Fatture F ON C.ID_cliente=F.id_cliente)
   INNER JOIN Unisci_fat_art U ON U.id_fattura=F.numero
      GROUP BY C.ragione_sociale;
```

ragione_sociale	Totale articoli
computer compound	19
il computer	17
la casa del pc	63
nobis	27
uni computer	38

Vediamo ora come fare per visualizzare, per ogni articolo, la quantità venduta a ciascun cliente, utilizzando un criterio multiplo di raggruppamento. Il codice e il risultato della query sono riportati di seguito:

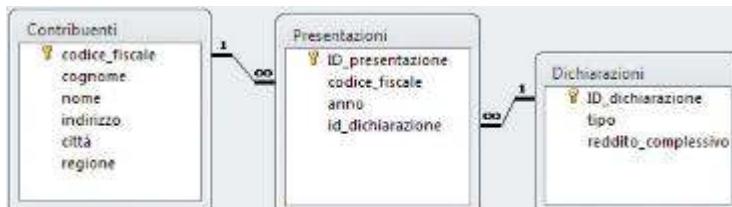
```
SELECT A.ID_articolo, C.ragione_sociale, SUM(U.quantità) AS Totale
FROM ((Clienti C INNER JOIN Fatture F ON C.ID_cliente=F.id_cliente)
   INNER JOIN Unisci_fat_art U ON U.id_fattura=F.numero)
      INNER JOIN Articoli A ON A.ID_articolo = U.id_articolo
         GROUP BY C.ragione_sociale, A.ID_articolo;
```

ID_articolo	ragione_sociale	Totale
1	computer compound	9
12	computer compound	10
1	il computer	4
2	il computer	12
5	il computer	1
2	la casa del pc	2
3	la casa del pc	6
4	la casa del pc	2
13	la casa del pc	10
14	la casa del pc	27
15	la casa del pc	2
16	la casa del pc	14
1	nobis	6
2	nobis	6
3	nobis	5
4	nobis	3
12	nobis	1
14	nobis	3
15	nobis	1
16	nobis	2
1	uni computer	2
2	uni computer	3
3	uni computer	23
4	uni computer	10

Limitazione delle tuple risultato

Per limitare le tuple risultato di una query è possibile utilizzare la clausola TOP seguita da un'espressione numerica "n", che specifica il numero di righe da restituire della query stessa, secondo la sintassi:

```
SELECT TOP "n" ... elenco campi
FROM ...
```

"n" identifica le prime righe che verranno visualizzate dalla query.

L'espressione TOP può anche essere utilizzata con le clausole INSERT, UPDATE e DELETE.

ESEMPIO

VALORE MAGGIORE TRA LE TUPLE RISULTATO

Prendiamo in esame il database Dichiarazione_redditi.mdb, che possiede la seguente struttura:

Vogliamo sapere in quale anno la somma del reddito complessivamente dichiarato è risultata più alta.

Per ottenere questo risultato, dobbiamo realizzare una query che applichi:

- una congiunzione (INNER JOIN) tra le tabelle coinvolte (Contribuenti, Presentazioni e Dichiarazioni);
- un raggruppamento (GROUP BY) per anno di presentazione;
- un operatore aggregato (SUM) al reddito complessivo;
- un ordinamento decrescente basato sull'operatore aggregato (ORDER BY ... DESC);
- una limitazione dei risultati alla sola prima riga (TOP 1).

Vediamo il codice completo della query:

```
SELECT TOP 1 P.anno,SUM(D.reddito_complessivo) AS [Reddito complessivo]
FROM (Contribuenti C INNER JOIN Presentazioni P ON
C.codice_fiscale=P.codice_fiscale)
INNER JOIN Dichiarazioni D ON P.id_dichiarazione=D.ID_dichiarazione
GROUP BY P.anno
ORDER BY SUM(D.reddito_complessivo) DESC;
```

Eseguendo la query, otteniamo una sola riga, che rappresenta la prima tupla dell'elenco ordinato in maniera decrescente:

anno	Reddito complessivo
2003	76730

ESEMPIO

RAGGRUPPAMENTO CON AUTO CONGIUNZIONE

Utilizziamo la tabella Stipendi, presente nel database Stipendi.mdb, la cui struttura è le seguente:

ID_impiegat	cognome	nome	id_superior	Stipendio
1	Marchionne	Giovanni	0	€ 4.200,00
2	Lupi	Pasquale	1	€ 2.345,00
3	Sandrone	Pippo	1	€ 678,00
4	Arlacchi	Samuele	1	€ 1.800,00
5	Lupone	Gino	0	€ 2.700,00
6	Soffice	Pamela	5	€ 2.500,00
7	Colotti	Marisa	5	€ 1.200,00
8	Marelli	Luca	5	€ 2.500,00
9	Simoni	Algise	5	€ 1.200,00
10	Carli	Elena	1	€ 800,00

Vogliamo calcolare lo stipendio medio degli Impiegati, raggruppati per superiore, con media dello stipendio maggiore di 1.500. Per questa operazione è necessario eseguire, oltre a un raggruppamento per cognome del superiore, anche una auto congiunzione. La codifica e il risultato della query sono riportati di seguito.

```
SELECT AVG(S1.stipendio) AS [Stipendio medio]
FROM Stipendi S1, Stipendi S2
WHERE S1.Id_superiore = S2.ID_impiegato
GROUP BY S1.id_superiore
HAVING AVG(S1.Stipendio)>1500;
```

Stipendio medio
€ 1.850,00

Come possiamo notare, prima di tutto lo stipendio del superiore non è stato conteggiato nel calcolo della media, in quanto nessun impiegato possiede il campo ID_impiegato uguale a id_superiore, cioè 0.

Per comprendere meglio, suddividiamo la query nelle singole fasi.

1 Vengono selezionate le tuple che soddisfano la condizione WHERE.

S1.ID_impiegato	S1.cognome	S1.nome	S1.id_superiore	S1.Stipendio	S2.ID_impiegato	S2.cognome	S2.nome	S2.id_superiore	S2.Stipendio
2	Lupi	Pasquale	1	€ 2.345,00	1	Marchionne	Giovanni	0	€ 4.200,00
3	Sandrone	Pippo	1	€ 678,00	1	Marchionne	Giovanni	0	€ 4.200,00
4	Arlacchi	Samuele	1	€ 1.800,00	1	Marchionne	Giovanni	0	€ 4.200,00
10	Carli	Elena	1	€ 800,00	1	Marchionne	Giovanni	0	€ 4.200,00
6	Soffice	Pamela	5	€ 2.500,00	5	Lupone	Gino	0	€ 2.700,00
7	Colotti	Marisa	5	€ 1.200,00	5	Lupone	Gino	0	€ 2.700,00
8	Marelli	Luca	5	€ 2.500,00	5	Lupone	Gino	0	€ 2.700,00
9	Simoni	Algise	5	€ 1.200,00	5	Lupone	Gino	0	€ 2.700,00

2 Viene effettuato il raggruppamento in base al campo S1.id_superiore.

3 Viene creata la target list, e per ogni gruppo di S1.id_superiore viene calcolata la media degli stipendi.

Stipendio medio
€ 1.405,75
€ 1.850,00

4 Come ultimo passaggio, vengono selezionate le sole tuple che soddisfano la condizione HAVING, cioè con valore AVG(S1.Stipendio) superiore a 1.500, con il risultato visto in precedenza.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Data la seguente tabella, indica quale, tra le query sotto riportate, calcola lo stipendio massimo e minimo per zona, escludendo dai calcoli coloro che non hanno stipendio.

ID	Cognome	Nome	Stipendio	id_zona
1	Martelli	Amelia		1
2	Colombo	Sonia	€ 1.700,00	2
3	Russo	Antonio	€ 1.850,00	2
4	Camelli	Alberto	€ 2.100,00	1
5	Simarco	Adele	€ 1.900,00	3
6	Lolli	Andrea		1
7	Dubini	Luca	€ 2.600,00	3
8	Simoncini	Gaia	€ 1.350,00	1

a
```
SELECT  T1.id_zona,  MIN(T1.Stipendio),
MAX(T1.Stipendio)
FROM Tabella T1, Tabella T2
WHERE T1.id_zona=T2.id_zona
GROUP BY T1.Cognome;
```
b
```
SELECT DISTINCT id_zona, MIN(Stipendio),
MAX(Stipendio)
FROM Tabella
WHERE Stipendio <> NULL;
```
c
```
SELECT id_zona, MIN(Stipendio),
MAX(Stipendio)
FROM Tabella
GROUP BY id_zona;
```
d
```
SELECT id_zona, MIN(Stipendio),
MAX(Stipendio)
FROM Tabella
WHERE MIN(Stipendio) <> NULL AND
MAX(Stipendio) <> NULL
GROUP BY id_zona;
```

2 Considerando la precedente tabella, indica quale, tra le seguenti query, determina la media degli stipendi per cognome e nome con somma maggiore di 2000 euro: (*2 risposte*)

a
```
SELECT id_zona, AVG(stipendio)
FROM Tabella
GROUP BY id_zona
HAVING AVG(Stipendio)>2000;
```
b
```
SELECT id_zona, AVG(stipendio)
FROM Tabella
GROUP BY Stipendio
HAVING AVG(Stipendio)>2000;
```
c
```
SELECT id_zona, AVG(stipendio)
FROM Tabella
WHERE AVG(Stipendio)>2000
GROUP BY Stipendio;
```
d
```
SELECT T1.id_zona, AVG(T2.stipendio)
FROM Tabella T1, Tabella T2
WHERE T1.id_zona=T2.id_zona
GROUP BY T2.Stipendio
HAVING AVG(T2.Stipendio)>2000;
```

3 Sempre con riferimento alla tabella dell'esercizio 1, quale tra le seguenti query permette di ottenere la media degli stipendi di coloro che guadagnano tra 1000 e 1700 euro?

a
```
SELECT AVG(Stipendio)
FROM Tabella
GROUP BY Stipendio
HAVING Stipendio BETWEEN 1000 AND 1700
```
b
```
SELECT AVG(Stipendio)
FROM Tabella
WHERE Stipendio BETWEEN 1000 AND 1700
```
b
```
SELECT AVG(Stipendio)
FROM Tabella
GROUP BY Stipendio
HAVING AVG(Stipendio) BETWEEN 1000 AND 1700
```
b
```
SELECT AVG(Stipendio)
FROM Tabella
WHERE AVG(Stipendio) BETWEEN 1000 AND 1700
```

4 Quale tra le seguenti frasi interpreta correttamente l'operatore AVG?

a Conta il numero di campi numerici NOT NULL
b Calcola la media aritmetica dei valori NOT NULL
c Calcola il massimo tra valori numerici
d Calcola la media aritmetica dei valori comprendendo anche valori NULL

5 Quale tra le seguenti frasi interpreta correttamente l'operatore STDDEV?

a Conta il numero di campi numerici NOT NULL
b Somma il contenuto di campi numerici
c Calcola la devianza standard di campi numerici
d Conta il numero di record anche non numerici

VERIFICA... le competenze

ESERCIZI

1 Apri il database Ordinativi.mdb, che contiene la tabella Ordinativi, con la struttura qui riportata.

Ordinativi
ID_ordine
Data_ordine
Importo
Cliente
Evaso

Crea le query necessarie per eseguire i seguenti compiti.
- Calcolo del rapporto tra importo massimo e minimo per ciascun cliente, con importi inferiori a 100.
- Conteggio degli ordini effettuati da ciascun cliente.
- Calcolo della media degli importi per data e per cliente.
- Calcolo della media degli importi per ciascun cliente.
- Calcolo dell'importo massimo per ordini evasi e non evasi.
- Conteggio di quanti ordinativi ci sono per ciascun cliente.
- Calcolo della media degli importi per ciascun cliente solo per importi maggiori di 50.
- Calcolo dell'importo massimo e minimo per data, solo se l'importo minimo calcolato è maggiore di 20.
- Calcolo, per ogni cliente, dell'importo minimo solo se è minore di 50.

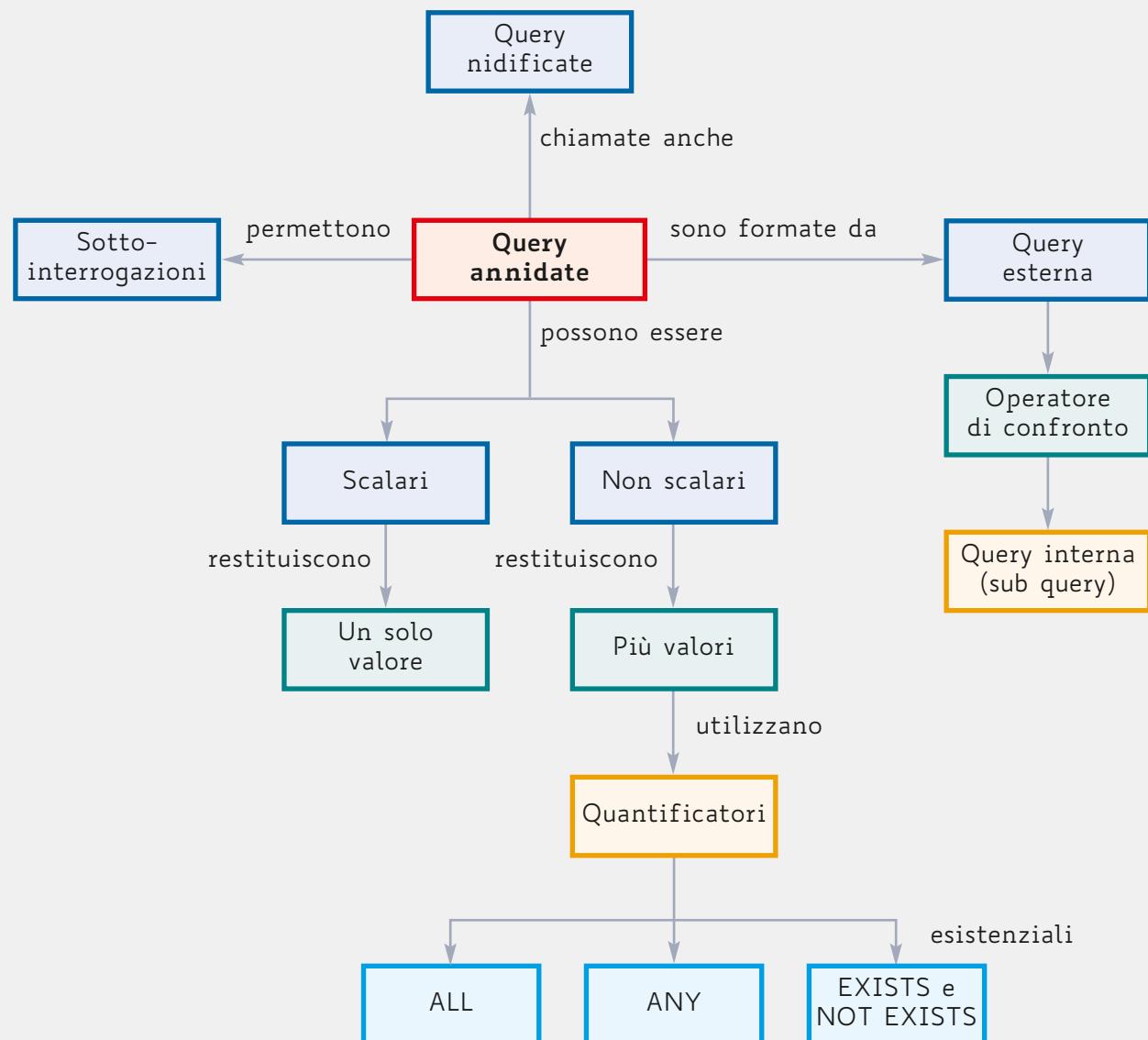
2 Apri il database Fatturazione.mdb, che possiede la seguente struttura:

Crea le query necessarie per eseguire i seguenti compiti.
- Elenco dei prezzi unitari degli articoli che sono stati venduti in più di 5 pezzi allo stesso cliente.
- Conteggio degli articoli venduti (SUM(quantità)) nelle fatture del mese ottobre (usa la funzione MONTH(10)).
- Calcolo del prezzo massimo e minimo per cliente, per i soli clienti di Bologna, Como e Milano.
- Conteggio di quanti articoli sono stati venduti a ciascun cliente.
- Calcolo del rapporto tra prezzo massimo e minimo per ciascun articolo.
- Calcolo della media del prezzo unitario degli articoli venduti al cliente "nobis".

3 Apri il database Dichiarazione_redditi.mdb, che possiede la seguente struttura:

Crea le query necessarie per eseguire i seguenti compiti.
- Calcolo della media di reddito complessivo dichiarato per i contribuenti della regione "Lombardia".
- Conteggio delle dichiarazioni effettuate per tipo "unico" relativo all'anno 2003.
- Elenco, per anno di presentazione, del massimo e minimo valore di reddito complessivo dichiarato.
- Calcolo dell'anno che ha registrato la somma di reddito complessivo dichiarato più bassa.
- Elenco di tutti i contribuenti con la relativa media di reddito dichiarata, anche per contribuenti che non avessero effettuato dichiarazioni.

VERIFICA... le conoscenze

ESERCIZIO

4 Definisci un database mediante codice SQL per memorizzare i dati relativi a tutte le canzoni che hanno partecipato al Festival di Sanremo. Per ciascuna canzone devi memorizzare il titolo, il testo, l'autore o gli autori e la durata in secondi. Assegna a ciascuna canzone un codice identificativo come chiave primaria. Per gli artisti che si sono esibiti al teatro Ariston devono essere registrati: il nome, il numero di CD venduti oltre a un codice come chiave primaria. Crea una relazione (n, n) tra le due tabelle, in modo che vengano registrati l'anno di partecipazione, la posizione ottenuta, il codice della canzone, i punti ottenuti, nonché un campo che specifichi se in quella edizione si tratta di un cantante cosiddetto "big". Definisci quindi le query che seguono.
- Elenco delle prime 10 canzoni della 50^{esima} edizione.
- Elenco dei primi 3 classificati di ogni edizione in ordine di importanza.
- Elenco di tutte le edizioni che hanno avuto più di 10 big.
- Elenco di quante edizioni hanno avuto complessivamente una media superiore ai 10 punti.
- Elenco di tutti gli autori di canzoni più lunghe di 3 minuti e mezzo, senza ripeterne i nomi.
- Calcolo della media di canzoni presenti per edizione.
- Elenco di tutte le canzoni che hanno una durata compresa tra 2'30" e 3'30".
- Elenco, contandole, di tutte le canzoni cantate nella 53^{esima} edizione.
- Calcolo della devianza standard dei punti relativi ai vincitori delle ultime 10 edizioni.
- Calcolo dell'autore che ha venduto di più e di meno in tutte le edizioni.
- Calcolo per ciascun autore delle canzoni scritte.
- Elenco dei cantanti e del numero di festival vinti.
- Elenco dei cantanti e del totale dei podi raggiunti.

5 In una scuola si vuole computerizzare l'orario delle lezioni settimanali. Per ciascuna classe e per ciascuna ora di lezione devono essere registrate la materia, il docente e il giorno della settimana. I docenti possono anche insegnare più materie. Per la soluzione del problema sono necessarie 5 tabelle. La tabella che contiene i dati anagrafici dei docenti prevede un codice identificativo per ciascun insegnante come chiave primaria. Le materie insegnate sono memorizzate in una tabella che possiede come chiave primaria l'identificativo di ciascuna materia. La tabella delle classi contiene, oltre all'identificativo della classe come chiave primaria, l'aula e il piano in cui è situata e il numero di alunni maschi e femmine. Inoltre la tabella orario permette di ottenere una relazione (n, n) tra la classe, il docente, la materia e il giorno della settimana. Per realizzare una relazione (n, n) tra la tabella Docenti e la tabella Materie è necessario introdurre una tabella di collegamento che contiene il codice del docente e della materia insegnata, associando a ciascun elemento un codice come chiave primaria. Definisci il database mediante codice SQL e dopo aver definito le tabelle con i vincoli relazionali corretti e popolato le stesse, definisci le query che seguono.
- Conto delle classi che hanno lezione di inglese al lunedì.
- Calcolo di quanti docenti insegnano chimica.
- Calcolo di quanti docenti insegnano italiano a classi più numerose di 25 alunni.
- Elenco di tutte le classi in ordine crescente.
- Elenco di tutte le discipline, calcolando quanti docenti insegnano ogni materia.
- Calcolo della media di alunni maschi per classe.
- Calcolo della media di alunni femmine per classe.
- Calcolo di quante classi hanno lezione di matematica all'ultima ora del sabato.
- Elenco dell'orario della classe 4ª B.

Le query annidate

IN QUESTA LEZIONE IMPAREREMO...

- a realizzare query annidate
- ad applicare i quantificatori ALL e ANY
- a comprendere il ruolo del quantificatore esistenziale

MAPPA CONCETTUALE

Query nidificate
chiamate anche
Query annidate
permettono
Sotto-interrogazioni
sono formate da
Query esterna
Operatore di confronto
Query interna (sub query)
possono essere
Scalari
restituiscono
Un solo valore
Non scalari
restituiscono
Più valori
utilizzano
Quantificatori
ALL
ANY
esistenziali
EXISTS e NOT EXISTS

Le query annidate

Query annidate scalari

Le query annidate scalari rendono le query dinamiche e rispondenti in tempo reale ai dati presenti nel database. Nel caso in cui le tuple risultanti dalla query interna siano più di una, appare un messaggio di errore: la query interna ha restituito più di una riga, rendendo così di fatto impossibile il confronto con un attributo della query esterna.

Le query annidate consentono di realizzare sotto-interrogazioni (query interne) annidate all'interno di altre query chiamate esterne. Permettono di combinare insieme più SELECT per ottenere un unico risultato. Le **query annidate scalari** producono dei risultati costituiti da una sola tupla con un solo attributo, utilizzabile ovunque sia assegnabile un valore scalare singolo.

Per eseguire due query in modo annidato (cioè una dentro l'altra) è necessario prima di tutto conoscere la sintassi generale delle SELECT nidificate.

External Query (Query esterna)	Operatore di confronto	Sub query (Query interna)
SELECT ... FROM ... WHERE espressione	= != > < !> !< >= <=	(SELECT ... FROM ... WHERE espressione)

Il funzionamento è abbastanza semplice: la sub query, che è indicata tra parentesi tonde, viene eseguita per prima; quindi, sulla base del risultato ottenuto, viene eseguita la query esterna. Tuttavia tale schema può essere protratto teoricamente all'infinito.

```
Query
   (Sub query)
       (Sub sub query)
         (Sub sub sub query)
            ...ecc...
```

Per comprendere meglio il ruolo delle sub query, utilizziamo il database PadriMadri.mdb, contenente le seguenti tabelle:

Maternità

cognome_figlio	nome_figlio	cognome_madre	nome_madre
Butti	Gino	Bari	Dina
Colombo	Patrizia	Bottari	Simona
Conti	Sara	Corelli	Lina
Conti	Gianna	Corelli	Lina
Corti	Loretta	Dini	Anna
Porta	Piera	Monti	Bianca
Valli	Mario	Tanzi	Mina
Valli	Paolo	Porta	Piera
Colombo	Piero	Fari	Sara

Tabella Maternità

Paternità

cognome_padre	nome_padre	cognome_figlio	nome_figlio
Butti	Gennaro	Butti	Gino
Colombo	Piero	Colombo	Patrizia
Conti	Andrea	Conti	Sara
Conti	Andrea	Conti	Gianna
Corti	Silvano	Corti	Loretta
Valli	Paolo	Valli	Mario
Porta	Pietro	Porta	Piera

Tabella Paternità

Vogliamo conoscere il nome della madre che ha un figlio con lo stesso nome del figlio di "Colombo Piero", sostanzialmente la moglie di "Colombo Piero". Per fare questo, utilizziamo una query interna, che cerca il nome del figlio di "Colombo Piero" nella tabella Paternità ("Patrizia") e una query esterna, che mostra il nome e cognome della madre che ha figli con il nome uguale a quello ottenuto nella query interna.

Se la query interna restituisse più valori, cioè se Colombo Piero avesse più di un figlio, si otterrebbe un errore, in quanto la query interna non sarebbe più di tipo scalare.

La query richiesta avrà la seguente sintassi:

```
SELECT Cognome_madre, Nome_madre
FROM Maternità
WHERE Nome_figlio =
(  SELECT Nome_figlio
    FROM Paternità
    WHERE Cognome_padre="Colombo" AND Nome_padre="Piero"
);
```

Query esterna

Query interna

Cognome_madre	Nome_madre
Bottari	Simona

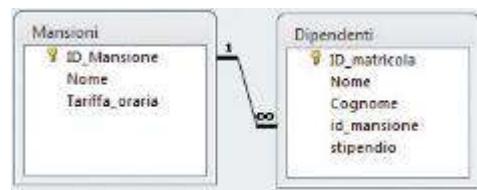
Il risultato della query è riportato a lato.

Per migliorarne la leggibilità, è consigliabile indentare sempre le query più interne.

Sempre con riferimento al database PadriMadri.mdb, vogliamo ottenere i nomi dei padri che hanno avuto meno figli di "Corelli Lina". Per fare questo, prima di tutto analizziamo la sub query che calcola il numero di figli di "Corelli Lina" dalla tabella Maternità. La query esterna, invece, calcola il numero di figli per padre, dalla tabella Paternità: mediante la clausola HAVING seleziona solo le tuple aventi un totale inferiore a quello calcolato nella sub query.

```
SELECT Cognome_padre, Nome_padre, COUNT(*) AS figli
FROM Paternità
GROUP BY Cognome_padre, Nome_padre
HAVING COUNT(*) <
(  SELECT COUNT(*)
   FROM Maternità
   WHERE Cognome_madre="Corelli" AND Nome_madre="Lina"
);
```

Cognome_padre	Nome_padre	figli
Butti	Gennaro	1
Colombo	Piero	1
Corti	Silvano	1
Porta	Pietro	1
Valli	Paolo	1

La nidificazione può essere effettuata anche con SELECT contenenti più di una condizione. Per esempio, sempre con riferimento al database PadriMadri.mdb, per conoscere il nome e cognome della nonna da parte di padre di "Valli Mario", selezioniamo nella query più esterna, dalla tabella Maternità, la madre che ha nome e cognome del figlio uguali ai risultati di due diverse sub query. La prima sub query seleziona il cognome e la seconda il nome del padre di "Valli Mario".
L'aspetto problematico deriva dal fatto che la query più esterna deve confrontare due attributi, il cognome e il nome. In tal caso, è necessario scrivere due query interne: nella prima viene confrontato il cognome, mentre nella seconda il nome.

```
SELECT Cognome_madre AS [Cognome nonna], Nome_madre AS [Nome nonna]
FROM Maternità
WHERE Cognome_figlio =
(  SELECT Cognome_padre
   FROM Paternità
   WHERE Cognome_figlio="Valli" AND Nome_figlio="Mario"
)
AND Nome_figlio =
(  SELECT Nome_padre
   FROM Paternità
   WHERE Cognome_figlio="Valli" and Nome_figlio="Mario"
);
```

Cognome nonna	Nome nonna
Porta	Piera

ESEMPIO

SUB QUERY

Utilizziamo il database Mansioni.mdb (che contiene le tabelle riportate a lato) per conoscere il nome dei dipendenti con stipendio superiore alla media dello stipendio degli autisti.

Per ottenere questo risultato, nella query interna calcoliamo la media dello stipendio dei dipendenti con mansione di autista, che confrontiamo con il campo Stipendio della query esterna. Vediamo il codice completo e il risultato ottenuto.

```
SELECT Cognome, Nome, Stipendio
FROM Dipendenti
WHERE Stipendio >
(   SELECT AVG(D.Stipendio)
    FROM Dipendenti D, Mansioni M
    WHERE M.ID_mansione=D.id_mansione
    AND M.Nome="Autista");
```

Cognome	Nome	Stipendio
Mario	Carli	2200
Giancluca	Corsi	6500
Arturo	Russo	2100
Caterina	Cabello	2800

Vogliamo ora conoscere il cognome e il nome di tutti i dipendenti che hanno stipendio inferiore alla media di quelli con mansione 9, ma superiore alla media di quelli con mansione 10. Per ottenere questo risultato, dobbiamo utilizzare SELECT contenenti più di una condizione. Vediamo il codice completo e il risultato ottenuto.

```
SELECT Cognome, Nome, Stipendio
FROM Dipendenti
WHERE Stipendio <
(   SELECT AVG(Stipendio)
    FROM Dipendenti
    WHERE id_mansione=9)
AND Stipendio >
(   SELECT AVG(Stipendio)
    FROM Dipendenti
    WHERE id_mansione=10);
```

Cognome	Nome	Stipendio
Mario	Carli	2200
Arturo	Russo	2100

Adesso ricerchiamo invece le mansioni che hanno una media di stipendio superiore alla media complessiva. Per ottenere questo risultato, calcoliamo nella sub query la media complessiva degli stipendi di tutti i dipendenti. Nella query esterna utilizziamo la INNER JOIN tra Mansioni e Dipendenti, il raggruppamento tramite il Nome della mansione e la clausola HAVING. Vediamo il codice completo e il risultato ottenuto.

```
SELECT M.Nome, AVG(D.Stipendio) AS media
FROM Mansioni M INNER JOIN Dipendenti D
ON D.id_mansione = M.ID_mansione
GROUP BY M.Nome
HAVING AVG(D.Stipendio) >
(   SELECT AVG(stipendio)
    FROM Dipendenti);
```

Nome	media
Manager	2200
Segretaria	2800
Top manager	6500

Query non scalari

Vediamo adesso come realizzare interrogazioni con sub query che restituiscono più di un valore. Per fare questo si utilizzano i quantificatori ALL, ANY, che vengono usati in combinazione con un operatore relazionale (=, >, <, <>, >=, <=).
Il quantificatore ALL indica che la condizione deve essere valida "per una qualsiasi tupla" delle righe risultanti dalla sub query, mentre ANY indica "tutte le tuple" risultanti dalla sub query.

Nell'esempio seguente la condizione può essere letta come "*maggiore di almeno una tupla*" tra quelle restituite dalla sub query:

```
...query esterna
 > ANY
... sub query
```

Nel prossimo esempio, invece, la condizione può essere letta come "*diversa da tutte le tuple*" restituite dalla sub query:

```
...query esterna
 <> ALL
... sub query
```

Il quantificatore = ANY è equivalente al quantificatore IN, mentre <>ALL equivale a NOT IN.

ESEMPIO

QUANTIFICATORI ALL E ANY

Utilizzando il database Mansioni.mdb, supponiamo di voler conoscere i nomi dei dipendenti che guadagnano meno di 2000 euro e che hanno la stessa mansione di quelli che ne guadagnano di più. Il risultato si ottiene calcolando, con una nidificazione, le tipologie di mansione dei dipendenti che guadagnano meno di 2000 euro, confrontate con le mansioni di coloro che guadagnano più di 2000. La query seguente risulta però errata:

```
SELECT Cognome, Nome, Stipendio
FROM Dipendenti
WHERE Stipendio <=2000
AND id_mansione =
(   SELECT id_mansione
    FROM Dipendenti
    WHERE Stipendio>2000);
```

Restituisce infatti il seguente errore:

Questo avviene in quanto la sub query ha cardinalità maggiore di uno, quindi il confronto non può essere valido. L'errore si verifica perché la condizione tra le due query è di tipo "ambiguo": questa, infatti, deve valere per uno o per tutti gli scalari calcolati dall'interrogazione interna?

Utilizzando un quantificatore (ALL, ANY) viene risolta l'ambiguità ed è possibile confrontare una serie di tuple:

```
SELECT Cognome, Nome, Stipendio
FROM Dipendenti
WHERE Stipendio <=2000
AND id_mansione = ANY
(   SELECT id_mansione
    FROM Dipendenti
    WHERE Stipendio>2000);
```

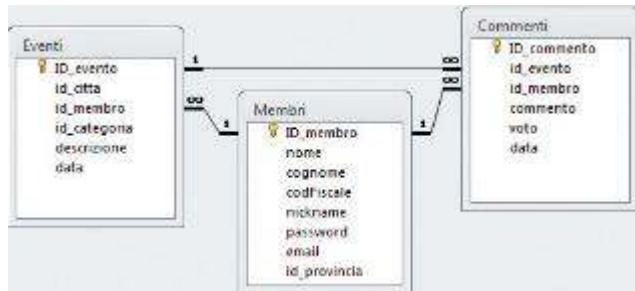
Cognome	Nome	Stipendio
Mario	Carletti	1800
Matteo	Pulici	1950

Vogliamo adesso conoscere il nome e la mansione dei dipendenti con uno stipendio più alto di quello di tutti i dipendenti con mansione "autista". In questo caso, è la stessa espressione "maggiore di tutti" a facilitare la comprensione del quantificatore da utilizzare, cioè ALL:

```
SELECT D.Cognome, D.Nome, M.nome AS mansione, D.Stipendio
FROM Dipendenti D INNER JOIN Mansioni M
ON M.ID_mansione=D.id_mansione
WHERE D.Stipendio > ALL
( SELECT D.stipendio
  FROM Dipendenti D INNER JOIN Mansioni M
  ON M.ID_mansione=D.id_mansione
  WHERE M.Nome="autista"
);
```

Cognome	Nome	mansione	Stipendio
Mario	Carli	Manager	2200
Gianclaudio	Corsi	Top manager	6500
Caterina	Cabello	Segretaria	2800
Adriana	Monti	Dirigente 1 liv	2500

Vogliamo infine elencare tutti i dipendenti che hanno la paga oraria superiore alla media della paga oraria degli impiegati di primo, secondo e terzo livello. Per ottenere questo risultato, si calcola la media della paga oraria riferita agli impiegati (mediante operatore Like), quindi la query esterna ricerca tutte le mansioni, con il relativo dipendente con paga oraria maggiore del risultato della sub query interna (> ALL):

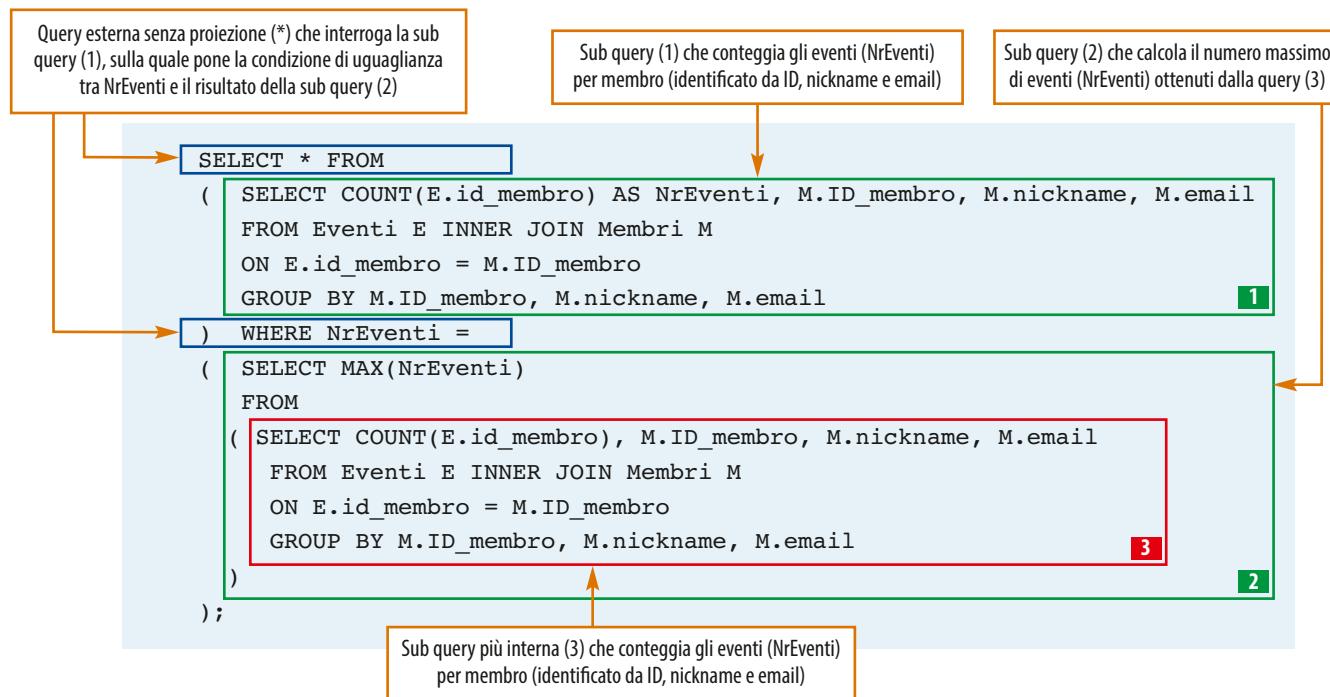
```
SELECT M.Nome AS mansione, D.Cognome, D.Nome,M.Tariffa_oraria
FROM Dipendenti D INNER JOIN Mansioni M
ON M.ID_mansione=D.id_mansione
WHERE M.Tariffa_oraria > ALL
(   SELECT AVG(Tariffa_oraria)
    FROM Mansioni
    WHERE Nome Like "impiegato*");
```

mansione	Cognome	Nome	Tariffa
Manager	Mario	Carli	90
Top manager	Gianclaudio	Corsi	140
Impiegato 3 liv	Gino	Alberti	18
Autista	Mario	Carletti	20
Autista	Matteo	Pulici	20
Autista	Arturo	Russo	20
Segretaria	Caterina	Cabello	22
Dirigente 1 liv	Adriana	Monti	31

Query complesse

Vediamo adesso delle query più complesse rispetto alle precedenti, che consentono di effettuare interrogazioni strutturate, rendendo in tal modo possibile la separazione di ogni singola parte significativa del comando SQL. Rappresentano un metodo alternativo alle congiunzioni per eseguire operazioni altrimenti non sempre eseguibili.

ESEMPIO

EVENTI E COMMENTI DI ALCUNI MEMBRI

Utilizziamo le tabelle Eventi, Commenti e Membri, contenute nel database MembriAssociazione.mdb, che rappresentano i commenti e gli eventi scritti da alcuni membri di un'associazione.

Vogliamo dapprima conoscere i nomi dei membri che non hanno mai inserito alcun commento. Le tabelle coinvolte sono Membri e Commenti. Per ottenere questo risultato, realizziamo una sub query che richiede l'elenco, con clausola DISTINCT per evitare nomi duplicati, di coloro che hanno scritto almeno un commento. La query esterna verifica, tramite il quantificatore <> ALL, che il nickname non sia presente tra i risultati della sub query:

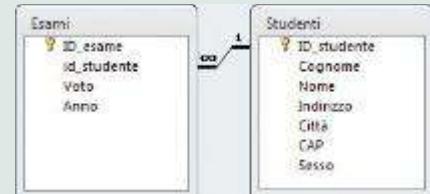
```
SELECT M.nickname
FROM Membri M
WHERE M.ID_membro <> ALL
( SELECT DISTINCT C.id_membro
  FROM Commenti C
)
ORDER BY M.nickname;
```

nickname
giogio
simonetta

Adesso vogliamo sapere quale membro ha scritto il maggior numero di eventi, quello cioè che possiede il numero massimo di tuple che lo rappresentano nella tabella Eventi. La query seguente risulta assai complessa, in quanto utilizza una query interna (3) che conteggia gli eventi per membro, una sub query (2) che calcola il numero massimo di eventi e una sub query (1) che conteggia gli eventi per membro. La query esterna effettua una SELECT per confrontare il risultato della sub query (1) con il risultato della sub query (2).

```
SELECT * FROM
( SELECT COUNT(E.id_membro) AS NrEventi, M.ID_membro, M.nickname, M.email
  FROM Eventi E INNER JOIN Membri M
  ON E.id_membro = M.ID_membro
  GROUP BY M.ID_membro, M.nickname, M.email
) WHERE NrEventi =
( SELECT MAX(NrEventi)
  FROM
  ( SELECT COUNT(E.id_membro), M.ID_membro, M.nickname, M.email
    FROM Eventi E INNER JOIN Membri M
    ON E.id_membro = M.ID_membro
    GROUP BY M.ID_membro, M.nickname, M.email
  )
);
```

Il risultato che si ottiene è il seguente:

NrEventi	ID_membro	nickname	email
4	4	toni	toni@live.it

PER SAPERNE DI PIÙ

IL QUANTIFICATORE ESISTENZIALE

Il quantificatore esistenziale permette di verificare l'esistenza (EXISTS) o non esistenza (NOT EXISTS) di almeno una tupla nel risultato di una sub query. La sintassi è la seguente:

```
SELECT attributo1
FROM Tabella1
WHERE [NOT] EXISTS
(   SELECT *
    FROM Tabella2);
```

Se una sub query restituisce almeno una tupla, EXISTS vale True, altrimenti False. Facciamo un esempio utilizzando le tabelle del database Studenti.mdb riportate a lato.

Vogliamo conoscere l'elenco degli studenti che non hanno superato alcun esame. Per ottenere questo risultato, si crea un'interrogazione esterna che elenca tutti gli studenti con proiezione sui dati anagrafici, mentre la congiunzione prevede l'uso del quantificatore esistenziale negato con NOT, che significa "*non esiste nemmeno una tupla nella sub query*". In questo esempio è stata usata la correlazione per legare tra loro le tabelle Studenti (nella query esterna) ed Esami:

```
SELECT ID_studente, Cognome, Nome
FROM Studenti S
WHERE NOT EXISTS
(   SELECT *
    FROM Esami
    WHERE S.ID_Studente=id_Studente);
```

ID_studente	Cognome	Nome
4	Corti	Giuseppe
5	Mancuso	Arturo

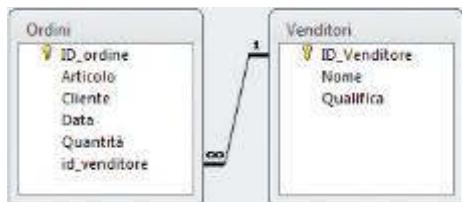
Se vogliamo conoscere quali sono gli studenti che hanno sostenuto almeno un esame, si utilizza il quantificatore EXISTS. Riportiamo di seguito la codifica e il risultato:

```
SELECT ID_studente, Cognome, Nome
FROM Studenti S
WHERE EXISTS
(   SELECT *
    FROM Esami
    WHERE S.ID_Studente=id_Studente)
```

ID_studente	Cognome	Nome
1	Verdi	Maria
2	Marelli	Lina
3	Russo	Antonia

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Quale quantificatore equivale a = ANY?

a IN
b NOT IN
c EXISTS
d NOT EXISTS

2 Quale affermazione riguardante la seguente query è corretta?

```
SELECT P.nome
FROM Persone P
WHERE P.ID =
( SELECT P.ID
 FROM Persone P
 WHERE P.nome="Mario" );
```

a La query è errata in quanto la sub query è scalare
b La query è corretta: effettua l'elenco delle persone che si chiamano "Mario"
c La query è errata in quanto la sub query non è scalare
d La query è corretta: effettua l'elenco delle persone che non si chiamano "Mario"

3 Date le tabelle Ordini e Venditori, indica quale query elenca i nomi dei venditori che non hanno venduto nulla nell'anno 2018.

a
```
SELECT V.nome FROM Venditori V
WHERE NOT EXISTS (SELECT * FROM Ordini O
WHERE O.id_venditore=V.ID_venditore AND
year(O.Data)<>'2018')
```
b
```
SELECT V.nome FROM Venditori V
WHERE V.ID_venditore <>ALL (SELECT
O. id_venditore FROM Ordini O, Venditori V1
WHERE O.id_venditore=V1.ID_venditore AND
year(O.Data)='2018')
```
c
```
SELECT V.nome FROM Venditori V
WHERE V.ID_Venditore <>ALL (SELECT
O. id_venditore FROM Ordini O
WHERE year(O.data)='2018')
```
d
```
SELECT V.nome FROM Venditori V
WHERE year(O.data)=2018
AND NOT EXISTS (SELECT * FROM
Venditori V INNER JOIN Ordini O ON
O. id_venditore=V.ID_venditore)
```

4 Quale affermazione riguardante la seguente query è corretta?

```
SELECT P.nome
FROM Persone P
WHERE NOT EXISTS
( SELECT P.ID
 FROM Persone P
 WHERE P.eta>25 );
```

a La query è errata in quanto il quantificatore è esistenziale
b La query è corretta: effettua l'elenco delle persone che non hanno più di 25 anni
c La query è errata in quanto la sub query è scalare
d La query è corretta: effettua l'elenco delle persone che hanno più di 25 anni

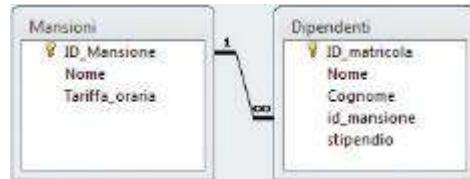
5 Quale quantificatore equivale a <> ALL?

a EXISTS
b NOT IN
c NOT EXISTS
d IN

6 Quale tra i seguenti è un quantificatore esistenziale?

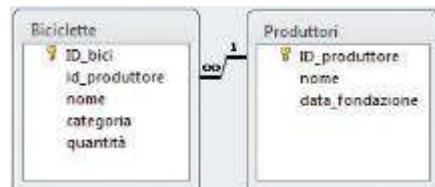
a <> EXISTS
b = ANY
c <> ALL
d NOT EXISTS

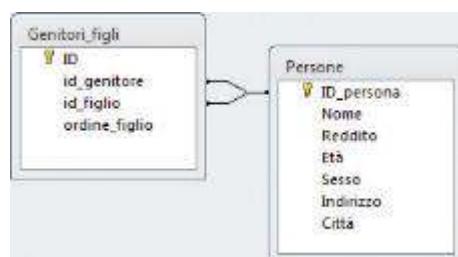
VERIFICA... le competenze

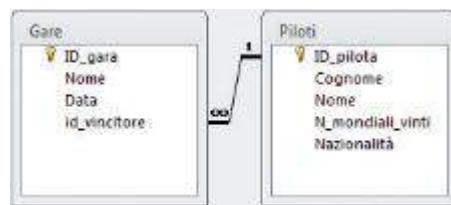
AREA DIGITALE

ESERCIZI

1 Apri il database Mansioni.mdb, che possiede la struttura riprodotta a lato.

Crea le query necessarie per eseguire i seguenti compiti.

- Indicazione del nome dei dipendenti che guadagnano più dell'autista che guadagna di meno.
- Elenco dei dipendenti che guadagnano meno di 2000 euro con la stessa mansione di quelli che ne guadagnano di più.
- Indicazione del dipendente che guadagna di meno tra quelli con stipendio superiore allo stipendio massimo degli impiegati di primo livello.
- Elenco di tutti i dipendenti che hanno paga oraria superiore alla media della paga oraria degli impiegati di primo, secondo e terzo livello.
- Elenco dei dipendenti che hanno stipendio superiore alla media della propria categoria.

Confronta le tue soluzioni con quelle contenute nel database Mansioni_solux.

2 Apri il database Biciclette.mdb, che possiede la struttura riprodotta a lato.

Crea le query necessarie per eseguire i seguenti compiti.

- Elenco dei produttori che hanno somma di biciclette prodotte inferiore alla media complessiva di biciclette prodotte.
- Elenco delle biciclette realizzate da produttori che hanno realizzato almeno un modello "tandem".
- Indicazione della categoria di bicicletta che è stata realizzata in numero maggiore rispetto a tutte le altre.
- Elenco delle biciclette prodotte in quantità totale superiore alla media di biciclette da "corsa" realizzate.

3 Data la struttura delle tabelle qui riprodotte, crea le query necessarie per eseguire i compiti di seguito indicati.

- Elenco dei genitori i cui figli guadagnano (tutti) più di 10000.
- Elenco, per ogni persona, dell'età media dei figli.
- Elenco dei figli la cui età è maggiore della media delle età dei figli dello stesso genitore.
- Elenco, per ogni persona, del primo figlio.
- Elenco delle persone senza figli.

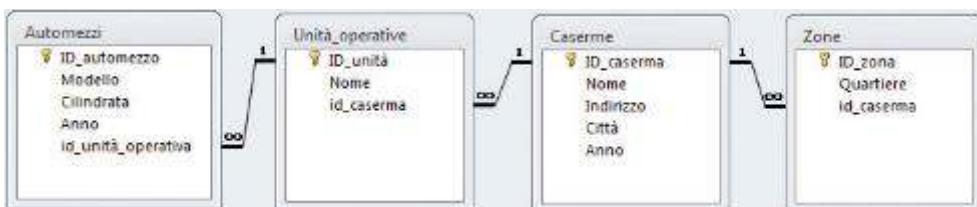
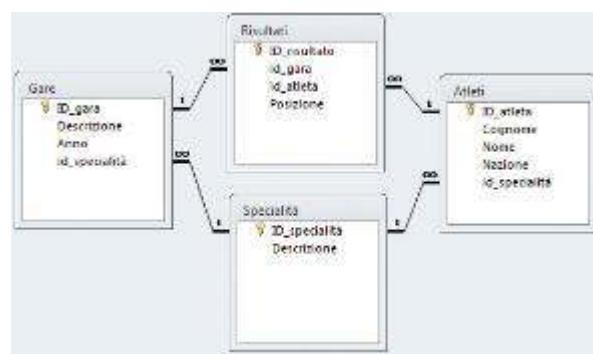
4 Data la struttura delle tabelle qui riprodotte, crea le query necessarie per eseguire i compiti di seguito indicati.

Gare: ID_gara, Nome, Data, id_vincitore — Piloti: ID_pilota, Cognome, Nome, N_mondiali_vinti, Nazionalità

- Elenco dei nomi dei piloti che, a partire dal 1970, hanno vinto almeno 2 gare.
- Elenco del nome di tutti i piloti che hanno vinto almeno 1 mondiale, ma non hanno mai vinto il Gran Premio di San Marino.
- Indicazione di quale pilota ha vinto più GP del Brasile.
- Elenco dei piloti che non hanno mai vinto nessun mondiale pur avendo vinto almeno una gara.
- Indicazione del pilota italiano che ha vinto più gare della media di tutti i piloti tedeschi.

VERIFICA... le competenze

ESERCIZI

1 Crea le seguenti tabelle:
Impiegati: (**ID**, Nome, Mansione, Data_Assunzione, Stipendio, Premio_P, id_azienda)
Aziende: (**ID**, Nome, Indirizzo, id_regione)
Regioni: (**ID**, Nome, abitanti)
Crea le seguenti query di selezione:
- Seleziona tutti gli impiegati che hanno premio di produzione maggiore di 1000 € della Lombardia e del Piemonte.
- Seleziona tutti gli impiegati assunti prima dell'anno 2015 in regioni con meno di 3.000.000 di abitanti.
- Calcola quanti impiegati ci sono per ciascuna azienda, solo per aziende con almeno 100 impiegati.
- Seleziona tutti gli impiegati che sono ingegneri, visualizzane l'azienda e la regione e ordinali per regione.
- Calcola la media dello stipendio degli ingegneri e dei dottori.
- Calcola lo stipendio massimo, regione per regione, solo degli impiegati che non hanno ricevuto un premio di produzione.

2 Data la seguente struttura:

Esegui le seguenti query SQL:
- Visualizza gli automezzi presenti per unità operativa.
- Visualizza le caserme che hanno mezzi più vecchi di 10 anni.
- Visualizza il numero di automezzi per unità operativa con cilindrata maggiore di 3000 cc.
- Visualizza le caserme e le unità operative che possiedono macchine a schiuma (modello) e quali quartieri possono essere raggiunti brevemente.
- Visualizza quanti automezzi per caserma hanno scale più alte di 18 metri.
- Visualizza, per le caserme che hanno scale più alte di 18 metri, il totale di metri di scale (campo lunghezza della tabella Automezzi) posseduti per zona, solo per zone con più di 60 metri totali di scale.

3 Data la seguente struttura:
Esegui le seguenti query SQL:
- Visualizza tutti i podi della specialità scherma.
- Visualizza tutti gli atleti che hanno ottenuto più di un podio.
- Visualizza la media di podi ottenuti per nazione.
- Visualizza le discipline (descrizione della gara) con gli stessi medagliati in olimpiadi diverse.
- Visualizza il totale di atleti medagliati per nazione e per olimpiade.

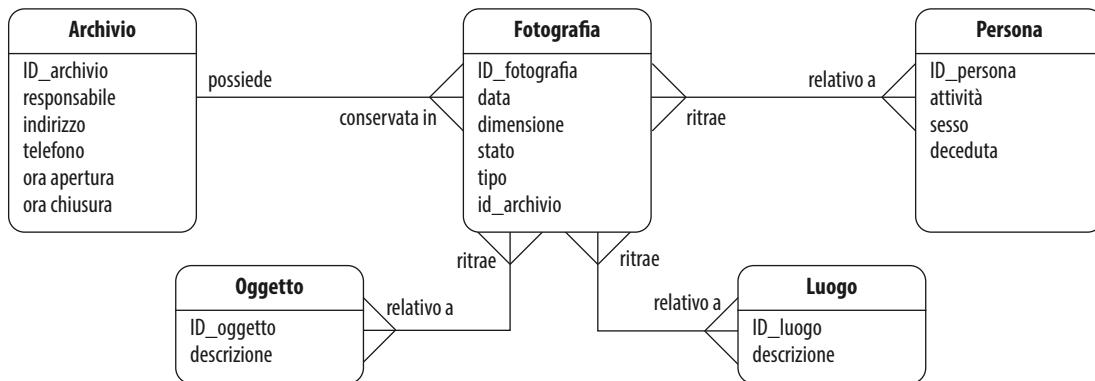
Verifica di fine Unità

ALTERNANZA SCUOLA-LAVORO

Scheda progetto n. 1

GESTIONE DI UN ARCHIVIO FOTOGRAFICO

PROBLEMA

Durante l'alternanza ti viene chiesto di collaborare con lo staff informatico per realizzare un database per la gestione di un archivio fotografico, formato da varie fotografie, disponibili in sedi diverse, di cui è noto il responsabile, l'indirizzo, il numero telefonico e l'orario di apertura e che rappresentano personaggi, oggetti o opere d'arte. Per ogni foto è nota la dimensione, il tipo (BN o colori), se su carta opaca o lucida, lo stato di conservazione e la data in cui è stata scattata. Per ogni personaggio sono noti i dati anagrafici. Per ogni opera d'arte è noto il titolo, l'artista che la ha realizzata, il luogo dove risiede e l'anno di realizzazione. Per gli oggetti il nome dell'oggetto stampato. Più foto possono rappresentare il medesimo oggetto, personaggio, opera d'arte o luogo.

Prerequisiti e obiettivi formativi

- Saper progettare un database.
- Conoscere il linguaggio SQL.

Indicazioni per la progettazione

Le fotografie sono registrate in base a un catalogo di soggetti possibili, e ciascun soggetto ha una propria chiave. Si creano così 8 tabelle per gestire la situazione. Le tabelle primarie contengono i dati relativi ai Luoghi, agli Oggetti e alle Persone. Mediante 3 tabelle di collegamento, si realizza una relazione n a n con la tabella Fotografie. Tali tabelle contengono soltanto due chiavi esterne, una alla tabella primaria e l'altra alla tabella Fotografie. In ultimo si crea la tabella Archivi, che è in relazione 1 a n con la tabella Fotografie.

Traccia per la realizzazione

Prima di tutto si creano le tabelle primarie, che corrispondono alle entità forti, cioè senza vincoli di chiavi esterne: Luoghi, Persone, Oggetti e Archivi. Si crea quindi la tabella Fotografie, con chiave primaria sulla singola foto e chiave esterna sull'archivio in cui è tenuta e quindi le tabelle di "rapporto" o di collegamento per risolvere la relazione n a n denominate rispettivamente Unisci_luoghi, Unisci_oggetti e Unisci_persone.
In collaborazione con lo staff, prepara le query che rispondano ai quesiti che ti verranno proposti.

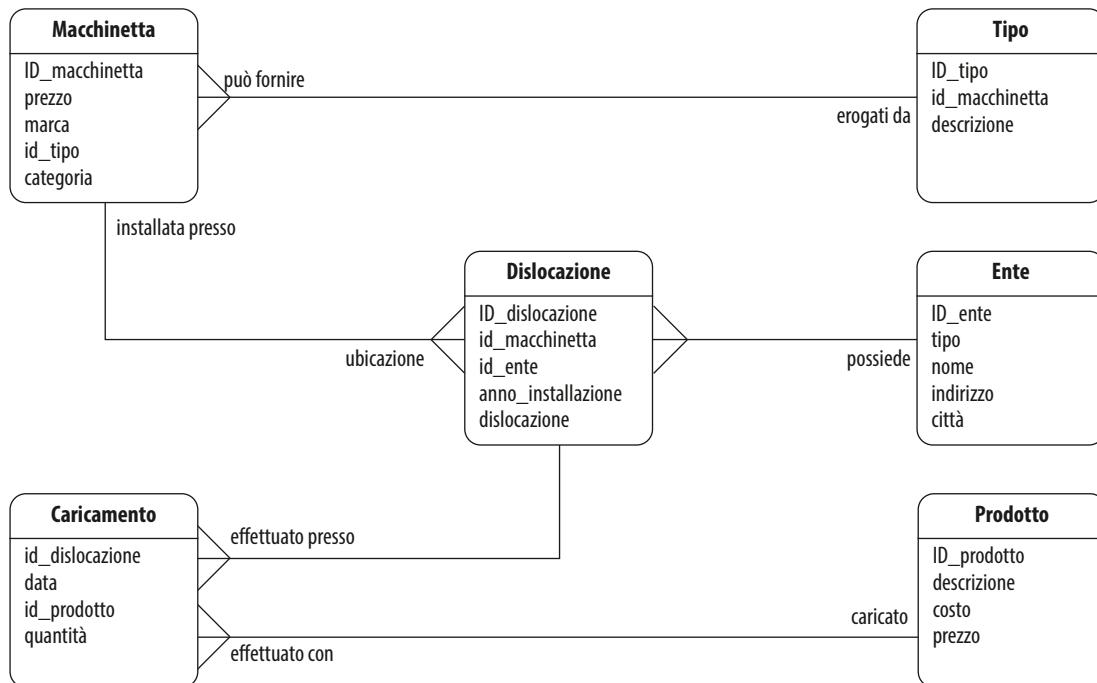
Preparazione e presentazione dei risultati

La soluzione del problema prevede la creazione delle tabelle del database, con i vincoli relativi. Il database verrà poi usato per la raccolta e memorizzazione dei dati riguardanti le fotografie nelle varie tabelle. Occorre fare una simulazione completa con un campione di ogni tabella e interrogazione per verificare la funzionalità del sistema e riportare in un documento osservazioni e proposte di modifica e/o miglioria del progetto.

ALTERNANZA SCUOLA-LAVORO

Proposta operativa n. 1

GESTIONE DISTRIBUTORI AUTOMATICI

PROBLEMA

Una azienda di gestione di distributori automatici vuole gestire mediante computer le proprie macchine dislocate in vari uffici, enti, scuole, ospedali ecc. Le macchine distributrici si dividono in diverse categorie, a seconda della tipologia di alimenti che erogano. Esistono macchinette che erogano bevande calde, bevande fredde, merendine, oppure un insieme di alcune di esse. Per ciascuna macchinetta esiste un codice di identificazione. Ciascuna macchinetta, a seconda della tipologia può erogare diversi prodotti, che vengono caricati periodicamente. Per valutarne i consumi è necessario memorizzarne i prodotti caricati e la data per ogni macchinetta.

Ti viene chiesto di realizzare un database per gestire i dati della situazione descritta.

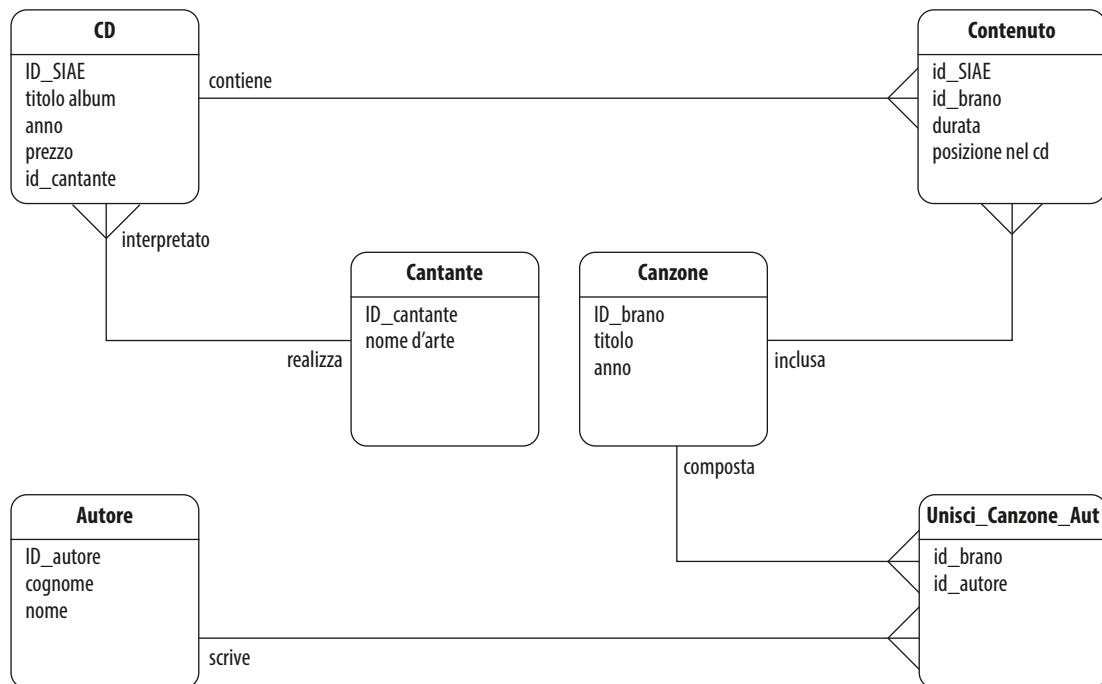
Passi operativi

- Crea la struttura relazionale che implementi il modello E-R sotto riportato.
- Elenca tutte le strutture che hanno più di una macchinetta.
- Calcola quanti enti possiedono almeno 4 macchinette di bevande fredde.
- Calcola le quantità di prodotti venduti per ciascun ente.
- Elenca le macchinette presenti presso gli enti della città di Milano.
- Calcola quanti prodotti sono stati caricati nell'anno 2018.

ALTERNANZA SCUOLA-LAVORO

Proposta operativa n. 2

GESTIONE CD AUDIO

PROBLEMA

Si vogliono gestire i prodotti discografici (CD) in vendita presso i negozi di musica. Per fare questo, occorre innanzitutto affermare che ogni CD in vendita contiene dei brani musicali. Per ciascun brano devono essere memorizzati i dati dell'autore o compositore e del cantante. Più autori possono concorrere alla stesura dello stesso brano, così come più brani possono essere stati creati dallo stesso autore.

Ti viene chiesto di realizzare un database per gestire i dati della situazione descritta.

Passi operativi

- Crea la struttura relazionale che implementi il modello E-R sotto riportato.
- Elenca le canzoni scritte da autori che iniziano per "I".
- Elenca i titoli dei CD che contengono canzoni di cui non si conosce l'anno di registrazione.
- Elenca i brani del CD con numero di serie 587T57P, ordinati per numero progressivo, con indicazione degli interpreti per ogni canzone.
- Calcola quanti brani sono stati scritti nel 2018.

CLIL

Key concepts

SQL Language

The SQL (*Structured Query Language*) is a language used to query a database. Its syntax is rigorous, but it is computationally incomplete.
It consists of queries, each ending with a semicolon.

Queries are made up of commands and clauses, and can belong to different sublanguages: DDL, DML and QL:
– DDL (*Data Description Language*) allows you to describe data by means of tables and related relationships;
– DML (*Data Manipulation Language*) allows you to modify both the content and the structure of tables;
– QL (*Query Language*) allows you to perform searches.

Searches can be carried out by means of selections, projections and joins.
Joins can be internal or external.

Using queries with aggregation operators, it is possible to do statistic calculations on the tables.

In case complex searches are required, it is possible to create queries within queries, using the nested queries technique.

Nested queries can be scalar (when they return only one value) or non-scalar (when they return more than one tuple).

The operator used for scalar subqueries is a traditional one (>, <, <>, >=, <=, Like, Between), while in the case of non-scalar queries the operator can also be ALL, ANY, EXISTS and NOT EXISTS.

Quiz

1 In DDL language, PRIMARY KEY is:
a the attribute
b the foreign key
c the relational relationship
d the domain

2 Which command among those listed below deletes all the record of the AddressBook table?
a DELETE AddressBook
b DELETE FROM AddressBook
c DELETE ALL FROM AddressBook
d DELETE FROM AddressBook WHERE PRIMARY KEY IS NULL

3 Which definition among those listed below identifies the rows of a table?
a Tuple
b Fields
c Attributes
d Schema

4 Which of the following constructs allows you to modify the structure of a table?
a INSERT
b DROP
c UPDATE
d ALTER

Programmazione lato server con php

UNITÀ 4

CONOSCENZE

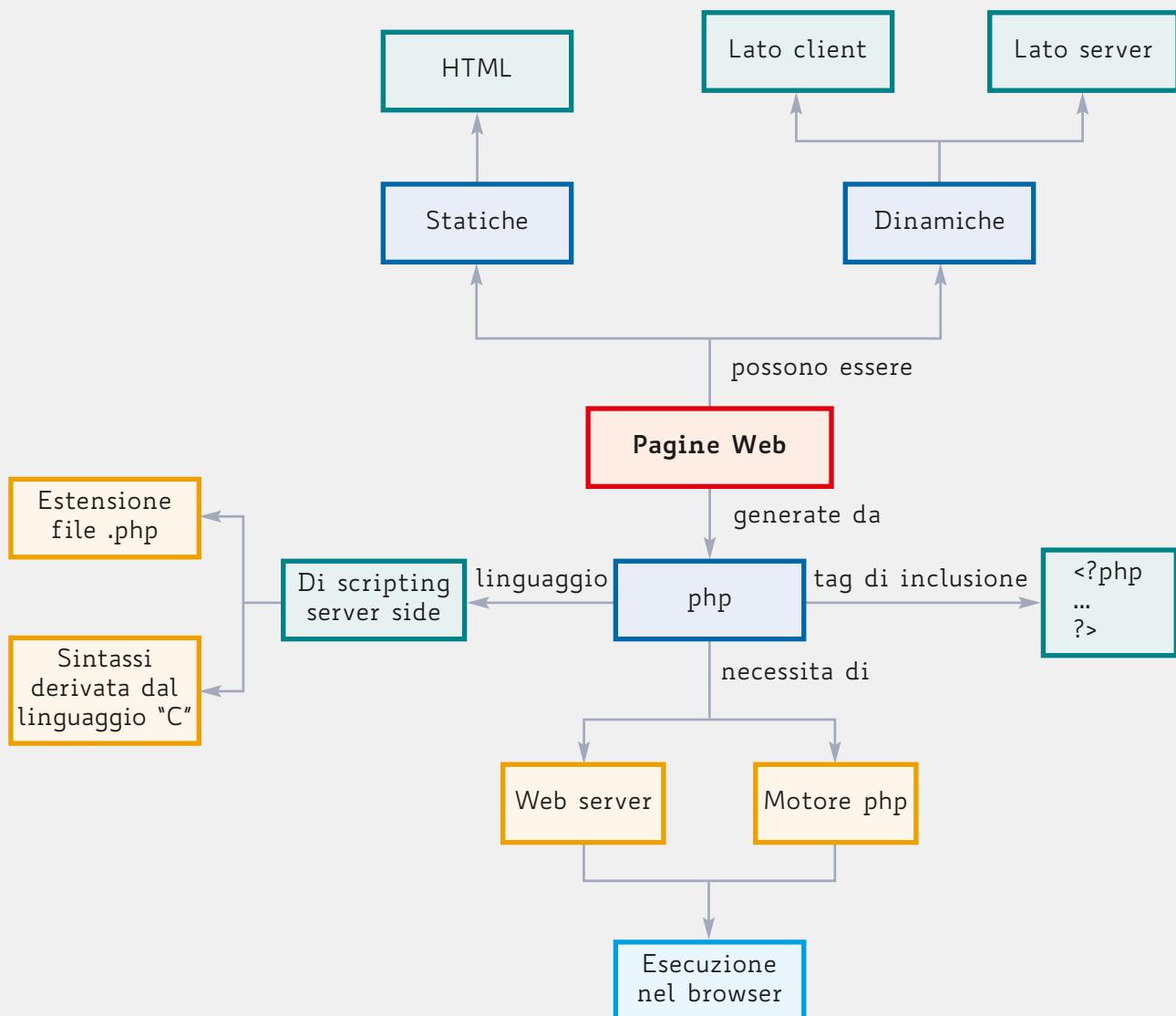
- Riconoscere le differenze tra script lato server e lato client
- Comprendere il ruolo della comunicazione client/server in http
- Identificare i principali elementi di uno script php
- Comprendere il ruolo dei Form nella programmazione php
- Individuare i principali elementi provenienti dai Form
- Conoscere la sintassi php
- Comprendere la tecnica postback

COMPETENZE

- Saper interrogare Access attraverso connessione da php
- Saper interrogare MySQL attraverso connessione da php
- Realizzare script contenenti connessioni ai database
- Realizzare script con Form e postback
- Realizzare pagine php persistenti
- Applicare le istruzioni php agli script

ABILITÀ

- Applicare le istruzioni php
- Utilizzare le istruzioni per realizzare script di gestione tabelle
- Realizzare script che utilizzino Form, sessioni
- Applicare cookie e sessioni alla persistenza
- Creare script di gestione array, file e tabelle di database

Il file relativi a esempi, esercizi e a eventuali soluzioni di questa unità sono contenuti nel CD-ROM e sono anche scaricabili dal sito hoepliscuola.it

La sintassi php

IN QUESTA LEZIONE IMPAREREMO...

- il concetto di pagina statica, dinamica e script lato server
- la sintassi del linguaggio php
- ad applicare i cicli

MAPPA CONCETTUALE

didattica inclusiva

HTML
Lato client
Lato server
Statiche
Dinamiche
possono essere
Pagine Web
generate da
Estensione file .php
Di scripting server side
linguaggio
php
tag di inclusione
<?php
...
?>
Sintassi derivata dal linguaggio "C"
necessita di
Web server
Motore php
Esecuzione nel browser

Pagine Web statiche o dinamiche

Le pagine che formano il Web possono essere statiche oppure dinamiche: sono statiche quando presentano gli stessi contenuti a tutti gli utenti che vi accedono, mentre sono dinamiche quando possono mostrare contenuti diversi in relazione alle azioni effettuate dagli utenti stessi (per esempio, il login di accesso a una sezione riservata di un sito).

Quando un client effettua una richiesta a un server per ottenere una pagina statica (per esempio con estensione .htm o .html), vengono eseguiti i seguenti passaggi:

– il client richiede una pagina al Web server (request);
– il Web server invia la pagina Web al client sotto forma di documento HTML (response);
– il browser del client interpreta e visualizza il documento HTML ricevuto.

Script
Uno script è un codice di programma sorgente che viene interpretato ed eseguito. Si possono avere script lato client, che vengono interpretati ed eseguiti dal browser, e script lato server. In questo caso il server, sulla base delle istruzioni di scripting, confeziona una pagina HTML e la invia al client: php (*Hypertext Preprocessor*) è un linguaggio di scripting lato server (server side).

Per quanto riguarda invece la richiesta di una pagina dinamica (request), il Web server, prima di inoltrarla al client, la trasforma in una pagina statica, generando al volo (on-the-fly) il codice HTML (che è il risultato di una elaborazione del codice dinamico presente nella pagina sul server stesso) e la restituisce all'utente (response): il contenuto di questa pagina può variare in relazione alla richiesta e alla elaborazione che è stata eseguita sul server da parte dello **script**.

Il codice sorgente della pagina dinamica è nascosto all'utente, che riceve sempre una pagina in formato HTML.

Quando un client effettua una richiesta a un server per ottenere una pagina dinamica (per esempio con estensione .php), vengono eseguiti i seguenti passaggi:
– il client richiede una pagina al Web server (request) (**1**);
– il Web server elabora il codice interno alla pagina (**2**), ed eventualmente effettua una richiesta al database, il quale restituisce le informazioni richieste (**3-4**), quindi sostituisce il codice sorgente php con codice HTML;
– l'interprete php restituisce la pagina al Web server (**5**);
– il Web server invia la pagina al client sotto forma di documento HTML (response) (**6**).

AREA DIGITALE

Le applicazioni tipiche degli script lato server sono le interrogazioni ai database remoti, per ottenere servizi come per esempio motori di ricerca, blog, forum di discussione.
In genere, le applicazioni per Internet sono la combinazione di tre tipologie di pagine, secondo una architettura posta su 3 livelli (*Three-Tier Architecture*), in cui il livello più basso è quello che riceve l'utente finale (front end), in formato DHTML (*Dynamic HTML*), il livello intermedio (middleware) corrisponde al formato delle pagine php, mentre il livello più alto (back end) corrisponde alle applicazioni che gestiscono i database remoti.

Il linguaggio php

php è un linguaggio che viene interpretato dal motore php (php engine), chiamato Zend. La sintassi prevede che l'inizio dello script venga marcato dal tag di apertura script (<?php). Quando il motore php incontra il tag di apertura script esegue tutto il codice, fino a quando non trova il tag di chiusura dello script, rappresentato da (?>).

È possibile anche "fondere" codice php all'interno di codice HTML: questa caratteristica prende il nome di HTML embedded: tutto ciò che si trova all'esterno dei tag che delimitano il codice php viene lasciato inalterato, mentre tutto quello che si trova all'interno dei tag <?php viene eseguito e i risultati, espressi in formato HTML, vengono inviati in risposta al browser che aveva richiesto la pagina.

L'esempio che segue mostra la sintassi necessaria per inserire codice php:

```
<?php
echo "<b>Salve Mondo</b>";
?>
```

L'istruzione echo visualizza sullo schermo il testo della stringa posta tra virgolette (" "); può essere incluso anche del codice HTML, in questo caso rappresentato dal tag del grassetto <b>.
La sintassi php è case sensitive. Inoltre, prevede che le istruzioni terminino sempre con il punto e virgola (;). I commenti si scrivono in stile C, cioè con doppio slash a inizio riga (//).

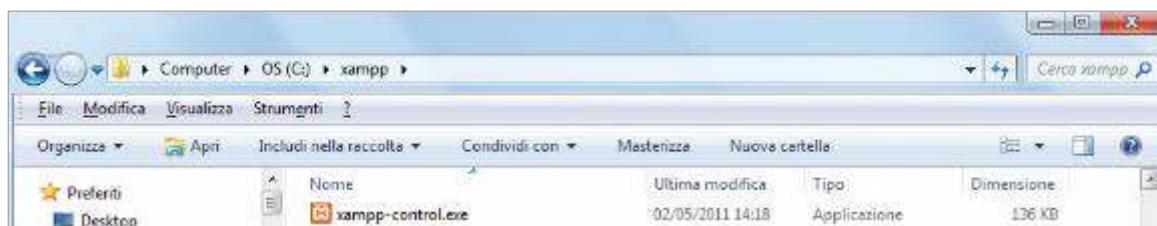
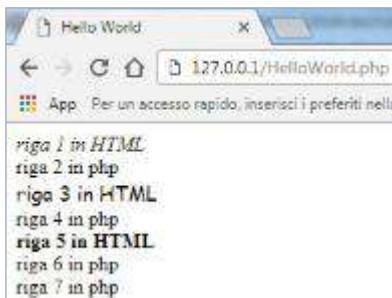
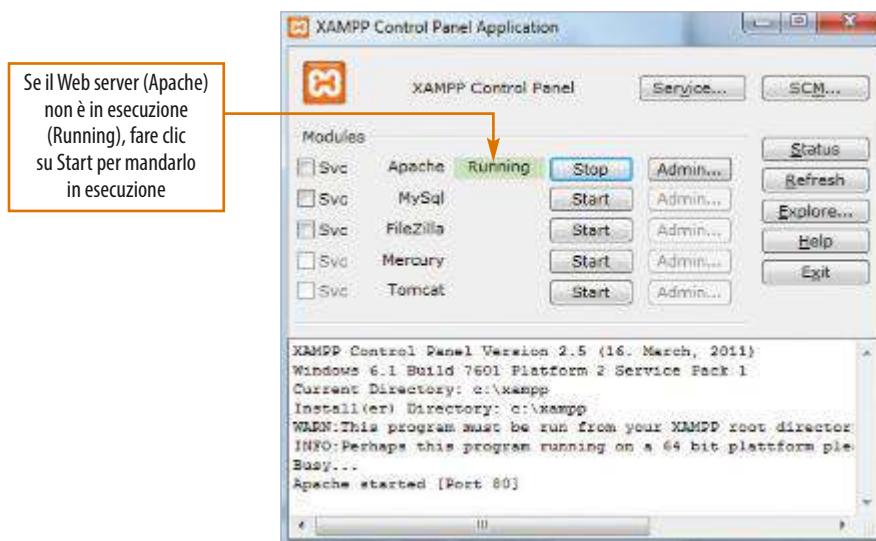
La notazione che utilizzeremo nel nostro testo, utile soprattutto per i neofiti, prevede la digitazione dei comandi HTML in maiuscolo, quando inseriti all'interno di istruzioni php. Questo, anche se non è aderente alle regole del Web, rende il codice molto più leggibile e semplice da comprendere.

Gli script php possono essere collocati sia nella <HEAD> che nel <BODY> di un documento HTML, tuttavia non possono iniziare nella <HEAD> per terminare nel <BODY>. Il file nel suo complesso viene eseguito dall'alto verso il basso, ed è possibile passare più volte da istruzioni HTML a istruzioni php.

Nell'esempio che segue mostriamo come creare una pagina utilizzando un editor in formato testo (in questo caso SciTE, scaricabile gratuitamente da www.scintilla.org).

Per eseguire tutti gli esempi che verranno proposti in questa unità di apprendimento dobbiamo avere installato sia il Web server che il motore php, oppure un pacchetto, come per esempio XAMPP o EasyPHP.
Il file deve essere collocato all'interno di una particolare directory (document root), che per XAMPP è rappresentata da htdocs:

```
c:\XAMPP\htdocs
```

ESEMPIO

ESEMPIO DI PROVA
Il codice mostra una semplice pagina nella quale abbiamo inserito del codice php all'interno di altro codice HTML: possiamo notare che, in questo caso, codice php e HTML si alternano.

```
1  <HTML>
2  <HEAD>
3  <TITLE>Hello World</TITLE>
4  </HEAD>
5  <BODY>
6  <I>riga 1 in HTML</I><BR>
7  <?php
8  echo "riga 2 in php<BR>";
9  ?>
10 <FONT FACE="comic sans MS">riga 3 in HTML</FONT><BR>
11 <?php
12 echo "riga 4 in php<BR>";
13 ?>
14 <B>riga 5 in HTML<BR>
15 <?php
16 echo "</B>riga 6 in php<BR>";
17 echo "riga 7 in php";
18 ?>
19 </BODY>
20 </HTML>
```

Per eseguire il programma è necessario salvarlo nella directory c:\xampp\htdocs, quindi mandare in esecuzione il pannello di controllo (xampp-control.exe) di XAMPP, presente nella directory di installazione c:\xampp:

Appare la finestra seguente:

Per testare la nostra pagina dobbiamo aprire il browser, digitando l'URL della pagina php, in questo caso rappresentato da 127.0.0.1, seguito dal nome del file (HelloWorld.php).
Il risultato che otteniamo è riportato a lato.

L'indirizzo 127.0.0.1, chiamato anche indirizzo di loopback, rappresenta l'indirizzo fittizio del computer sul quale è in esecuzione il Web server e dovremo sempre indicarlo per eseguire le nostre pagine php.

Puoi trovare il codice di questo esempio nel file HelloWorld.php.

La sintassi di php

Nel linguaggio php le variabili possiedono un nome che inizia obbligatoriamente con il simbolo del dollaro ($) e sono case-sensitive, pertanto $variabile sarà diversa da $Variabile.

Gli operatori aritmetici sono gli stessi del linguaggio C: somma (+), sottrazione (–), moltiplicazione (*), divisione (/) e resto della divisione intera (%). A differenza del C, tuttavia, il risultato dell'operazione è sempre di tipo float, anche se la divisione avviene tra due interi.
L'operatore di assegnazione fondamentale è il segno di uguale (=) e può essere applicato anche nel modo seguente: a + = 3; corrisponde a a = a + 3;. Questo vale anche per tutti gli altri operatori aritmetici, oltre che per l'operatore di concatenamento di stringhe. Per esempio:

```
$stringa. = “cognome”;
```

equivale a:

```
$stringa = $stringa.”cognome”;
```

Gli operatori condizionali sono maggiore (>), minore (<), uguale (==), diverso (!=), maggiore o uguale (>=), minore o uguale (<=), quelli logici sono invece AND (&&), OR (||), NOT (!).

Le funzioni matematiche principali utilizzate in php sono riportate nella tabella che segue:

Funzione	Significato
ceil()	Arrotonda le frazioni all'intero superiore
floor()	Arrotonda le frazioni all'intero inferiore
is_nan()	Verifica se un dato valore non sia un numero
pow()	Espressione esponenziale
round()	Arrotonda un numero non intero
sqrt()	Radice quadrata
srand()	Inizializza il generatore di numeri casuali
rand()	Genera un numero casuale

Anche i costrutti principali derivano direttamente dal linguaggio C: il codice che segue mostra l'uso del costrutto di selezione if:

```
if (condizione)
{
   //vero
}
else
{
   //falso
}
```

La selezione multipla viene effettuata con il costrutto switch:

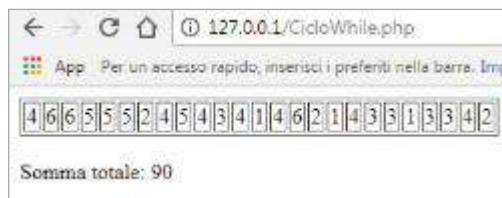
```
switch(variabile)
{
   case valore:
   {
       //azioni
       break;
   }
   case valore:
   {
       //azioni
       break;
   }
    …
   default:
   {
       //azioni
       break;
   }
}
```

L'iterazione possiede la seguente sintassi:

A controllo iniziale	A controllo finale
`while (condizione)`	`do`
`{`	`{`
`   //istruzioni`	`   //istruzioni`
`}`	`} while (condizione)`

ESEMPIO

CICLO WHILE

In questo esempio utilizziamo un ciclo while per simulare il lancio di un dado mentre la somma dei numeri generati è minore di 90.

```
1  <?php
2  $somma=0;
3  echo "<TABLE BORDER=1><TR>";
4  while ($somma<90)
5  {
6      //simulazione lancio dado con numero intero tra 1 e 6
7      $NumeroCasuale=rand(1,6);
8      echo "<TD>$NumeroCasuale</TD>";
9      $somma+= $NumeroCasuale;
10 }
11 echo "</TABLE>";
12 echo "<BR>Somma totale: $somma";
13 ?>
```

Lo script prima di tutto stampa, con l'istruzione echo, il codice HTML per la generazione di una tabella con bordi visibili (riga 3). Quindi inizia un ciclo while, che termina quando la variabile $somma non contiene più un valore inferiore a 90. All'interno del ciclo troviamo, nella riga 7, la generazione del numero casuale nella variabile $NumeroCasuale, che avviene tramite la funzione rand (1, 6). La variabile $somma viene usata per accumulare i numeri generati (riga 9). Nella riga 11 viene chiusa la tabella attraverso la scrittura con echo della stringa HTML e infine, nella riga 12, viene stampato il valore totale.

Eseguendo lo script, otteniamo il risultato seguente:

Puoi trovare il codice di questo esempio nel file CicloWhile.php.

L'iterazione con ciclo for possiede la seguente sintassi, anch'essa derivata dal linguaggio C:

```
for(inizializzazione; condizione; incremento)
```

Esiste anche il ciclo foreach, che consente di scandire gli elementi di un array e possiede la seguente sintassi:

```
foreach($nome_array as $chiave => $valore)
{
  // Codice da eseguire per ogni elemento dell'array
  // $chiave conterrà il valore della chiave dell'elemento i-esimo
  // $valore conterrà il valore dell'elemento i-esimo
}
```

Il controllo sul tipo di dati

È necessario tenere presente che i dati provenienti dal client potrebbero essere di un tipo non atteso a priori. A tal proposito, php mette a disposizione alcune funzioni in grado di verificare il tipo di dato presente in una variabile. Per "testare" il tipo di una variabile o di una espressione esistono le funzioni:
- is_int (espressione);
- is_float (espressione);
- isset (espressione);
- settype (espressione);
- gettype (espressione, tipo).

Esistenza variabili
Nel campo della programmazione lato server è necessario, a volte, verificare se il client ha inserito qualcosa in una variabile oppure l'ha lasciata vuota.

Casting esplicito
È una conversione esplicita e per effettuarla si deve indicare il tipo "destinazione" tra parentesi – (int), (bool), (float), (double), (string) – seguito dall'espressione da trasformare.

La funzione is_int verifica se la variabile contiene un numero intero, is_float verifica se contiene un dato numerico di tipo reale, isset verifica l'**esistenza di una variabile**, cioè se è stata inizializzata e non contiene "NULL".

La funzione gettype() permette di conoscere il tipo di una variabile o il tipo restituito da un'espressione, mentre invece settype() consente di forzare una variabile ad assumere il tipo indicato.

Le variabili di php non possiedono un tipo vero e proprio, ma il tipo viene assegnato in modo implicito, in base al primo dato a esse assegnato, oppure possono essere convertite mediante **casting esplicito**.

Vediamo un esempio di casting esplicito:

```
$n = 25.54;
$numero = (int) $n;                // converte in 25
$n = 10;
$numero = (float) $n;              // converte in float, $numero conterrà 10.0
```

Le stringhe

In php, una variabile stringa può essere inizializzata in 2 modi: con apice singolo ('), oppure con apice doppio ("). Le stringhe ad apici singoli sono assai limitate, in quanto non consentono l'inserimento di caratteri di escape (così detti perché iniziano con il simbolo di escape, che è \) e, nel caso di stampa di stringhe (p. es: echo 'stringa'), non è possibile inserirvi le variabili.

Per concatenare le stringhe si utilizza il carattere punto (.). Per esempio:

```
$a="infor";
$b="matica";
$c=$a . $b;
echo "le due stringhe concatenate producono:$c";        // informatica
```

Possiamo concatenare anche stringhe con numeri, in quanto il numero viene automaticamente convertito in stringa. Vediamo un esempio.

ESEMPIO

CONCATENAZIONE DI STRINGHE

Nel seguente codice viene richiesta la stampa di una espressione contenente sia stringhe che valori numerici:

```
1  <?php
2  $a = 23;                //Intero
3  $b = 11.5;              //Float
4  $c = 0.0;               //Float
5  $d = " ottengo: ";      //Stringa
6  $c = $a / $b;
7  echo "Se divido ".$a." con ".$b.$d.$c;
8  ?>
```

Il codice mostra una conversione automatica in stringa senza perdere alcun valore numerico, infatti il risultato è:

Puoi trovare il codice di questo esempio nel file Stringhe.php.

Per trasformare un tipo qualsiasi in stringa si può usare l'operatore di casting (string), oppure la funzione strval().

Invece, per ottenere il carattere ASCII da una stringa o viceversa, si usano le funzioni ord() e chr():

```
$a=ord("A");
echo $a . ' ';          // stampa 65
$a=$a+1;
echo $a . ' ';          // stampa 66
echo chr($a);           // stampa "B"
```

Vediamo un esempio.

ESEMPIO

STAMPA DEI CARATTERI ASCII

Si vogliono stampare i primi 255 caratteri ASCII mediante un ciclo for che genera una tabella. Mediante la funzione chr trasformiamo il numero intero presente nella variabile $i nel corrispondente codice ASCII:

```
1  <?php
2  echo "<TABLE WIDTH=30%><TR>";
3  //Iniziamo dal carattere 32 per evitare simboli non stampabili
4  for($i=32;$i<256;$i++)
5  {
6      //Stampa nuovo elemento in tabella
7      echo "<TD>".chr($i);
8      //Se i è divisibile per 16 andiamo a capo di una riga
9      if (($i%16)==0)
10         echo "<TR>";
11 }
12 ?>
```

Si ottiene in output il seguente risultato:

Puoi trovare il codice di questo esempio nel file Ascii.php.

AREA DIGITALE

VERIFICA... le conoscenze

AREA DIGITALE

Esercizi per l'approfondimento

SCELTA MULTIPLA

1 Il linguaggio php è di tipo:
- **a** solo interpretato
- **b** solo compilato
- **c** solo interpretato sul client
- **d** solo compilato sul client
- **e** solo interpretato sul server

2 Quale è la document root di default per XAMPP?
- **a** c:\programmi\apache\httpdoc
- **b** c:\xampp\htdocs
- **c** c:\programmi\htdocs
- **d** c:\programmi\easyphp\docs

VERO/FALSO

1. La funzione is_ seguita dal tipo permette di conoscere il tipo di una variabile ed è applicabile a qualsiasi variabile anche non ancora creata. V F
2. Il tipo di una variabile viene dichiarato in modo implicito all'atto della prima assegnazione. V F
3. php richiede un tipo esplicito durante la dichiarazione di una variabile. V F
4. L'operatore di casting per gli interi è (int). V F
5. Convertendo una stringa che contiene un numero intero in float si ottiene soltanto una stringa vuota. V F
6. Nelle stringhe a doppi apici non è possibile inserirvi le variabili. V F
7. La concatenazione di stringhe avviene mediante il carattere +. V F
8. La funzione chr() consente la conversione da carattere ASCII a codice numerico. V F

SIMULAZIONE INFORMATICA

1 Scrivi il risultato dei seguenti blocchi di istruzioni, sapendo che $x vale 5 all'inizio.

```
$a=12;
if ($a--)
    $x-=5;
else
    $x+=$a%2;
```

$a= (______)
$x= (______)

```
$a=10;
$x--;
$x+=$x--;
$x-=--$a;
```

$a= (______)
$x= (______)

```
$b=3;
$x=($x--)+($b++);
```

$x= (______)
$b= (______)

```
$c=10;
$s="c";
$c-=$$s++;
$b=0;
$x=(($b)?--
$c:++$b);
```

$b= (______) $c= (______)
$x= (______)

2 Scrivi il risultato dei seguenti blocchi di istruzioni, sapendo che $x vale 5 all'inizio.

```
$x=5;
$a=12;
while ($x!=10)
{
 if !($a<10)
 {
      $a-=2;
 }
 $x=$a;
}
```

$a= (______) $x= (______)

```
$x=6;
$a=11;
do
{
 if (!$a<10)
 {
      $a-=2;
 }
 $x+=$a;
}while ($x<=14);
```

$a= (______) $x= (______)

VERIFICA... le competenze

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Disegna una scacchiera di 20 x 20 caselle.

2 Visualizza una tabella di 20 righe e 20 colonne di colore blu con bordo=3 e all'interno un numero che aumenta di 3 ogni volta.

3 Mostra una tabella che contiene l'elenco delle squadre di calcio di serie A e a fianco la foto e poi il collegamento al sito delle società.

4 Visualizza i primi 100 numeri interi.

5 Mostra a video i primi *N* numeri del triangolo di Tartaglia.

6 Una variabile contiene una stringa a piacere, una seconda stringa contiene invece "123456789": lo script deve stampare un carattere della prima stringa indicato dall'elemento 3 della seconda stringa. Usa una sola istruzione per la stampa.

7 Calcola e stampa a video quanti secondi sono trascorsi tra due date inserite.

8 Calcola e stampa la somma dei quadrati di un numero *N*.

9 Calcola e stampa la distanza tra due punti del piano cartesiano.

10 Calcola e stampa le radici, dati i coefficienti di una equazione di secondo grado.

11 Scrivi una semplice pagina php che scriva "Benvenuto nel nostro sito" e verificane il funzionamento, salvandola con il nome prova.php.

12 Scrivi una seconda pagina e salvala col nome index.html. Verifica le differenze rispetto al caso precedente. Verifica che il client riceva solo il codice HTML di questa pagina e non il codice php.

13 Scrivi una pagina php che stampi a video il tuo nome ripetuto 10 volte separato da una linea verde.

14 Scrivi una pagina php che scriva 10 nomi a scelta, ogni volta di colore diverso.

Visibilità delle variabili e funzioni

IN QUESTA LEZIONE IMPAREREMO...

- a comprendere il campo di visibilità delle variabili
- a dichiarare e richiamare funzioni passando parametri

MAPPA CONCETTUALE

didattica inclusiva

define
definite con
Costanti
Simbolo $
precedute da
Variabili
Simbolo $$
precedute da
Variabili di variabile
utilizza
include file esterni con
include()
Linguaggio php
Funzioni
possono essere
Funzioni utente
Funzioni built in
Variabili d'ambiente
$_GET
$_PUT
$_FILES
$_SERVER
$_SESSION
$_COOKIE

Costanti e variabili d'ambiente

In php le costanti, a differenza delle variabili, devono essere sempre dichiarate. Una costante è formata da un identificatore (nome) e da un valore e viene creata con l'istruzione define.
Il codice seguente crea una costante che contiene il numero di pigreco:

```
define("PIGRECO",3.14);
echo PIGRECO;
```

A differenza delle variabili, le costanti non iniziano con il segno del dollaro ($) e vengono scritte in maiuscolo, per differenziarle dalle variabili.

AREA DIGITALE

La funzione phpinfo()

In php esistono anche particolari strutture dati predefinite, chiamate variabili d'ambiente o superglobals, che consentono di agevolare la programmazione coinvolgendo il server. Riportiamo nella tabella che segue quelle più importanti.

Variabile d'ambiente	Descrizione
$_GET	Contiene i campi HTTP ricevuti in GET
$_POST	Contiene i campi HTTP ricevuti in POST
$_FILES	Consente di effettuare l'upload di variabili
$_SERVER	Contiene informazioni riguardanti il server
$_COOKIE	Rappresenta i cookies HTTP
$_SESSION	Rappresenta le variabili di sessione

ESEMPIO

VISIBILITÀ DELLE VARIABILI

La visibilità delle variabili, o scope, è il contesto in cui vengono definite e sono visibili. Se una variabile viene definita al di fuori della funzione in cui è utilizzata, non sarà sempre visibile.

```
1  <?php
2  function prova_scope()
3  {
4      $a=true;
5      if ($a)
6          //$b viene inizializzata nella function e quindi possiede scope
7          //solo all'interno di prova_scope()
8          $b="attualmente";
9
10 }
11 //nel codice che segue $b non esiste
12 if (isset($b))
13     echo $b;
14 else
15     echo "variabile non visibile!";
16 ?>
```

La variabile $b possiede uno scope limitato alla funzione prova_scope in cui è stata definita. Quindi, al di fuori di questa funzione, non è visibile, cioè non esiste. Quando eseguiamo lo script otteniamo il messaggio seguente:

localhost/Visibile.php
App Per un accesso rapido, inserisci i preferiti nella
variabile non visibile!

Se invece dichiariamo la variabile $b di tipo globale, attraverso l'istruzione di assegnazione scritta al di fuori della funzione prova_scope, come indicato di seguito:

```
1  <?php
2  //variabile $b dichiarata globale quindi visibile in tutte le funzioni
3  $b="valore iniziale";
4  function prova_scope()
5  {
6      $a=true;
7      if ($a)
8          //$b viene usata nella function ma il suo scope è adesso globale
9          $b="attualmente";
10 }
11 //nel codice che segue $b esiste in quanto è adesso globale
12 if (isset($b))
13     echo $b;
14 else
15     echo "variabile non visibile!";
16 ?>
```

eseguendo lo script, otteniamo lo stesso valore che è stato usato per inizializzarla, cioè "valore iniziale", in quanto lo scope di $b non è più interno alla funzione, ma globale rispetto allo script.

Puoi trovare il codice di questo esempio nel file Visibile.php.

Variabili di variabile

Alcune volte può essere conveniente poter accedere a una variabile il cui nome è rappresentato da un'altra variabile di tipo stringa che lo contiene.
Normalmente, una variabile viene settata dal valore in esso contenuto, come in questo esempio:

```
$prima_parte = "infor";
```

La variabile $prima_parte contiene un testo e quindi è stata settata a tipo string. Se tuttavia ci fosse la necessità di accedere a una variabile che si chiama $infor, sarebbe necessario usare una variabile di variabile: per fare questo, basta anteporre un altro simbolo dollaro ($$), come mostrato nel codice che segue:

```
$infor=100;
$variabile="infor";
echo $$variabile;          // Viene stampato il contenuto di $infor, quindi 100
```

Vediamo un esempio.

ESEMPIO

VARIABILI DI VARIABILI

Utilizziamo le variabili di variabili per effettuare un ciclo in cui assegnare a un Form una serie di caselle di testo con un nome (nome1, nome2, nome3 ecc., fino a nome10) e un valore che è dato dal contenuto della variabile $nome1, $nome2 ecc.
All'interno del ciclo viene assegnata alla variabile $temp una stringa contenente la variabile $nome seguita dall'indice $i. L'istruzione successiva applica poi il concetto di variabile di variabile (riga 17).

```php
<?php
$nome1="Cognome:";
$nome2="Nome:";
$nome3="Indirizzo:";
$nome4="cap:";
$nome5="Città:";
$nome6="Regione:";
$nome7="Provincia:";
$nome8="Telefono:";
$nome9="Fax:";
$nome10="Codice:";
//creazione tag del Form
echo "<FORM ACTION='pippo.php'><TABLE WIDTH=35%><TR>";
for ($i=1; $i<11; $i++)
{
    $temp = "nome$i";
    $valore = $$temp;          //assegniamo a $valore il contenuto di $nome1
                               //$nome2 poi $nome3 ecc..
    echo "<TR><TD>".$valore."<TD><INPUT TYPE='text' NAME='nome$i' VALUE='$valore'><BR>";
}
?>
```

Si ottiene un Form con l'aspetto seguente:

Puoi trovare il codice di questo esempio nel file varVariabile.php.

Le funzioni utente

In php, le funzioni utente devono essere dichiarate o definite mediante la parola chiave function, seguita dal nome della funzione da creare. Le funzioni ricevono parametri (in entrata) e possono restituire altri parametri (in uscita).
Vengono richiamate mediante un criterio analogo a quello usato per le variabili, scrivendone semplicemente il nome.
La sintassi per la loro definizione è la seguente:

```
function nome (ev. parametri formali separati dalla virgola)
{
 //istruzioni
 return eventuale valore di ritorno;
 }
```

La sintassi per la loro chiamata è invece la seguente:

```
valore restituito = nome(ev. parametri attuali separati dalla virgola);
```

Le funzioni possono essere richiamate prima o dopo il blocco di definizione. Inoltre, le funzioni non possiedono alcun tipo di ritorno, il tipo viene infatti adeguato secondo la tecnica della conversione implicita.

php ammette tre tipologie di passaggio dei parametri alle funzioni:
– per default;
– per valore;
– per riferimento.

Vediamo di seguito un esempio che mostra come avviene il passaggio dei parametri per default.

ESEMPIO

PASSAGGIO PARAMETRI PER DEFAULT

La funzione indicata stampa il parametro passato, se tuttavia non riceve alcun valore stampa un valore predefinito:

```
1  <?php
2  //definizione funzione con parametro formale di default
3  function chiama($nome = "nessuno")
4  {
5      echo "il mio nome é: ".$nome;
6      echo "<BR>";
7      return;
8  }
9  //chiamata senza parametro
10 chiama();
11 //chiamata con parametro
12 chiama("testo di prova");
13 ?>
```

Il risultato è il seguente:

```
il mio nome è: nessuno
il mio nome è: testo di prova
```

Puoi trovare il codice di questo esempio nel file Parametri.php.

Nel passaggio parametri per valore, la funzione non modifica i parametri passati e all'uscita tali parametri non vengono restituiti modificati al segmento di codice chiamante.
Il passaggio parametri per riferimento (o indirizzo) consente invece a una funzione di restituire gli argomenti modificati. Per passare un argomento per riferimento a una funzione, si deve anteporre il simbolo di ampersand o e commerciale (&) al nome dell'argomento nella definizione della funzione.
Nel passaggio di parametri per indirizzo la modifica dei parametri all'interno della funzione influisce sui parametri attuali che sono stati passati.

Vediamo di seguito un esempio che illustra la differenza tra parametri passati per valore e parametri passati per riferimento.

ESEMPIO

PASSAGGIO PARAMETRI PER VALORE E PER RIFERIMENTO

Nel seguente codice possiamo notare che, mentre dopo la chiamata alla funzione passaValore() i valori vengono restituiti invariati, dopo la chiamata alla funzione passaIndirizzo() i valori vengono

invece restituiti incrementati di 100. Da notare inoltre che nell'intestazione della funzione passaIndirizzo() i parametri passati (per indirizzo) sono preceduti dal carattere &.

```
1  <HTML>
2  <HEAD>
3  <?php
4  function passaValore($numero1,$numero2)
5  {
6      $numero1=$numero1+100;
7      $numero2=$numero2+100;
8  }
9  function passaIndirizzo(&$numero1,&$numero2)
10 {
11     $numero1=$numero1+100;
12     $numero2=$numero2+100;
13 }
14 ?>
15 </HEAD>
16 <BODY>
17 <?php
18 $a=20;
19 $b=30;
20 echo "<HR>Prima della chiamata a passaValore<BR>";
21 echo "$a,$b <BR>";
22 passaValore($a,$b);
23 echo "Dopo la chiamata a passaValore<BR>";
24 echo "$a, $b <BR>";
25 echo "<HR>Prima della chiamata a passaIndirizzo<BR>";
26 echo "$a,$b <BR>";
27 passaIndirizzo($a,$b);
28 echo "Dopo la chiamata a passaIndirizzo<BR>";
29 echo "$a,$b <BR>";
30 ?>
```

Dopo l'esecuzione, otteniamo i seguenti risultati attesi:

```
Prima della chiamata a passaValore
20,30
Dopo la chiamata a passaValore
20, 30
-------------------------------------
Prima della chiamata a passaIndirizzo
20,30
Dopo la chiamata a passaIndirizzo
120,130
```

Puoi trovare il codice di questo esempio nel file PassaggioParametri.php.

Inclusione di codice da file esterno

Le funzioni possono essere definite indifferentemente nella <HEAD> o nel <BODY>, oppure ancora possono essere incluse da un file esterno con l'istruzione include().
L'inclusione è meglio che avvenga nella intestazione, per evitare problemi di visibilità e perché essa verrà caricata subito in memoria.
L'esempio che segue mostra come caricare una funzione presente in un file esterno, il quale può avere una qualunque estensione; preferibilmente, per uniformità, i programmatori php usano .inc.

ESEMPIO

INCLUSIONE DI FILE ESTERNO

La funzione disegnaTabella() è presente in un file esterno chiamato disegna_tab.inc, che viene richiamato attraverso il comando include.

Il file disegna_tab.inc è il file che contiene la funzione:

```
<?php
//definizione funzione che riceve come parametri le righe da creare
function disegnaTabella($righe,$colonne)
{
    echo "<TABLE WIDTH=30% BORDER=1>";
    for($r=0;$r<$righe;$r++)
    {
        echo "<TR>";
        for($c=0;$c<$colonne;$c++)
            echo "<TD> ";
    }
    echo "</TABLE>";
}
?>
```

Il programma che include il file esterno e utilizza la funzione è il seguente:

```
<?php
echo "Il disegno della tabella avviene includendo una funzione esterna<BR>";
// inclusione del file contenente la funzione disegnaTabella()
include("disegnatabella.inc");
//Chiamata della funzione
disegnaTabella(10,20);
?>
```

In questo caso, il risultato è una tabella di 10 righe per 20 colonne:

Puoi trovare il codice completo dell'esempio nel file DisegnaTab.php.

METTITI ALLA PROVA

• Applicazione del passaggio parametri • Utilizzo della funzione include()

Apri il file PassaggioParametri.php

1 Modifica il codice spostando la funzione che passa i parametri per indirizzo all'interno di un file esterno di nome funzione.inc.
2 Modifica il codice del file PassaggioParametri.php in modo tale che venga incluso il codice del file creato al punto 1.
3 Ripeti la stessa operazione anche per la funzione che passa i parametri per valore e verifica il funzionamento dello script.

VERIFICA... le competenze

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Crea una funzione che, se passati 2 numeri, restituisca la loro somma.

2 Modifica l'esercizio precedente in modo tale che il risultato venga passato sotto forma di 3° parametro alla funzione.

3 Crea una funzione che, passati il numero di righe e colonne, il colore di sfondo e lo spessore del bordo, crei una tabella con tali informazioni.

4 Crea una funzione che riceva 3 parametri, due numeri e la relativa operazione da effettuare (+ – * /). Se il parametro vale zero, restituisce "errore", altrimenti il risultato.

5 Dato un numero *N*, calcola con funzione ricorsiva la somma di tutti i numeri da 1 a *N*.

6 Crea una funzione che sottolinei una frase passata come parametro.

7 Crea una funzione che riceva una stringa e la trasformi in carattere maiuscolo e minuscolo in modo alternato, trasformando ogni singolo carattere.

8 Crea una funzione che stampi a video (in una tabella) i primi *n* numeri primi, con *n* ricevuto come parametro.

9 Crea una funzione che verifichi se una stringa immessa è palindroma o no.

10 Crea una funzione che calcoli l'ordinata di una funzione del tipo $y = f(x)$ passandole l'ascissa.

11 Crea una funzione ricorsiva in grado di calcolare il MCD di 2 numeri.

12 Calcola e stampa il prodotto tra due numeri interi e positivi tramite funzione ricorsiva sapendo che:
– se $b=0 allora $a*$b=0
– se $b>0 allora $a*$b = $a*($b-1)+$a

13 Crea una funzione che, se passato un parametro, ne calcoli il logaritmo in base 10, se due, il secondo viene elevato alla quarta potenza, se tre, il terzo deve essere elevato alla terza.

14 Crea una funzione che calcoli il MCD secondo il metodo euclideo.

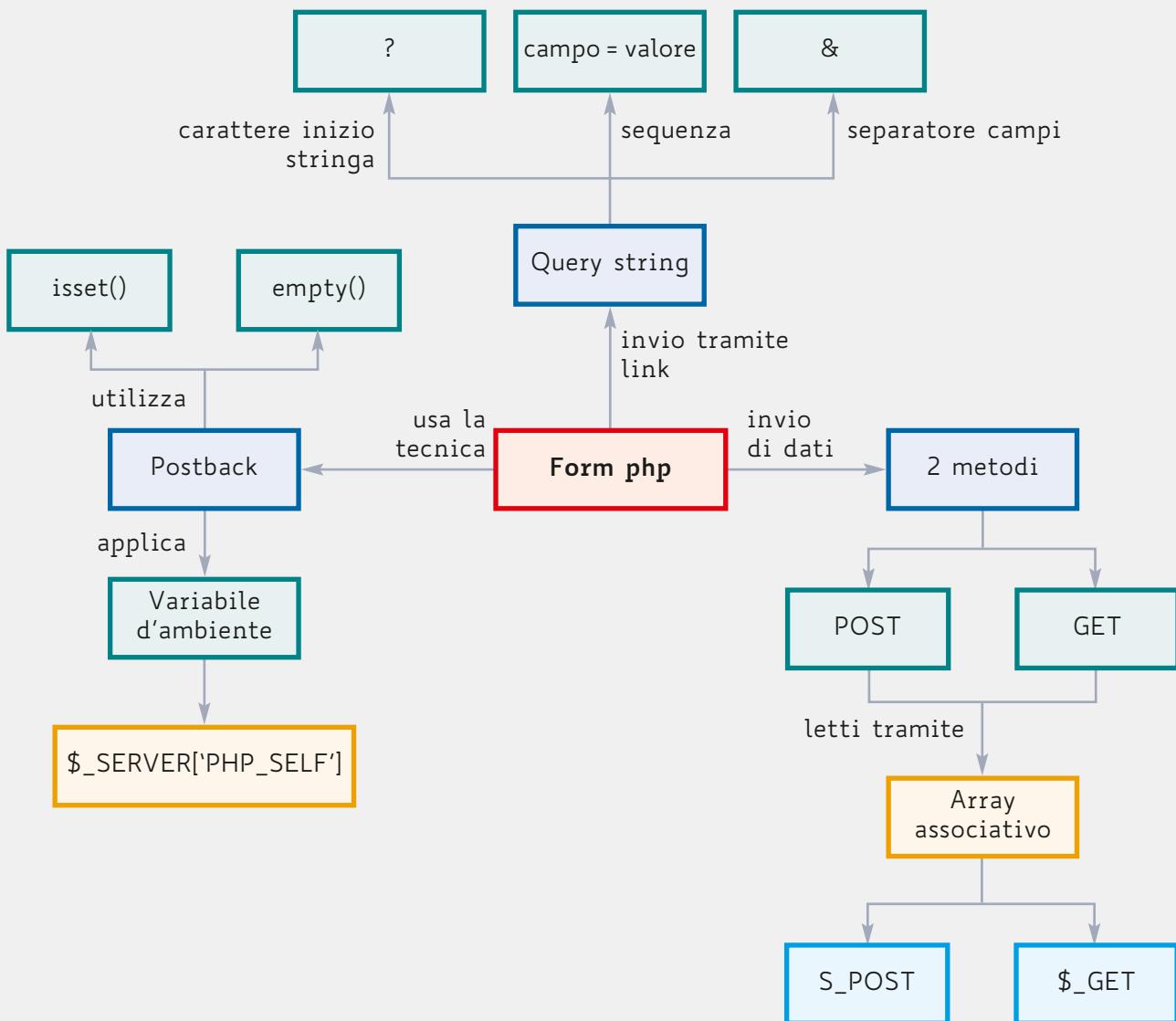
15 Crea una funzione che calcoli il minimo comune multiplo secondo l'algoritmo di Euclide.

16 Crea una funzione che mostri una immagine da un elenco di immagini presenti in una cartella in base al numero random passato come parametro.

17 Crea una funzione che riceva la base e l'esponente come argomenti e restituisca la potenza b^e.

18 Calcola e stampa a video la serie numerica seguente:

$a_1 = 3;$
$a_2 = 9;$
...
$a_n = 3 * a_{n-1} - a_{n-2}$ per *n* maggiore o uguale a 3

quindi 3, 9, 18, 45, 117 ecc.

La pagina prevede l'uso di una funzione ricorsiva per il calcolo del risultato. Tale funzione, di tipo ricorsivo, possiede come test di uscita *n* = valore immesso dall'utente.

I dati provenienti dai Form

IN QUESTA LEZIONE IMPAREREMO...

- a elaborare i campi GET e POST ricevuti dai Form
- a utilizzare la tecnica postback
- a elaborare i campi ricevuti da query string

MAPPA CONCETTUALE

?
campo = valore
&
carattere inizio stringa
sequenza
separatore campi
Query string
isset()
empty()
invio tramite link
utilizza
usa la tecnica
invio di dati
Postback
Form php
2 metodi
applica
Variabile d'ambiente
POST
GET
$_SERVER['PHP_SELF']
letti tramite
Array associativo
S_POST
$_GET

I dati inviati dai Form

I moduli di HTML (Form) possono inviare i propri campi a uno script php per la successiva elaborazione.
Per accedere al campo ricevuto da un modulo si utilizza l'array associativo $_POST, oppure $_GET, a seconda del metodo di invio usato.
Per esempio, per assegnare alla variabile $lettura il campo ricevuto in POST e in GET scriveremo:

```
// se il form usa il metodo POST
$lettura_post=$_POST['nome campo']
// se il form usa il metodo GET
$lettura_get=$_GET['nome campo']
```

GET e POST specificano come le informazioni contenute nel Form devono essere passate al server. GET indica al browser di inviare i valori inseriti nel modulo dall'utente all'indirizzo della richiesta, separandoli dal nome della pagina di destinazione con un punto di domanda. Tali valori vengono perciò inseriti in una stringa di query lunga al massimo 256 caratteri, visibile all'utente nella barra di indirizzo del browser e perciò poco sicura, soprattutto in caso di invio di informazioni riservate, come per esempio le password.
POST rende invece invisibile all'utente la stringa inviata al server in quanto viene inclusa nel corpo (sezione <BODY>) del messaggio.

Vediamo di seguito un esempio che illustra il funzionamento del metodo POST.

ESEMPIO

SOMMA TRA DUE NUMERI

L'esempio si compone di due pagine. La prima contiene il codice HTML necessario per mostrare un modulo contenente due caselle di testo e il pulsante di invio, il cui nome è Somma.html.
Mediante l'attributo ACTION=Somma.php viene indicato il percorso dello script che riceverà i campi modulo e provvederà alla loro elaborazione per fornire i risultati:

```
3      <TITLE>Pagina HTML contenente il FORM</TITLE>
4    </HEAD>
5  - <BODY>
6  - <FORM ACTION="Somma.php" METHOD="post">
7  -   <TABLE>
8  -     <TR><TD>Primo numero
9  -     <TD><INPUT TYPE="text" NAME="numero1">
10 -     <TR><TD>Secondo numero
11 -     <TD><INPUT TYPE="text" NAME="numero2">
12 -     <TR><TD><INPUT TYPE="submit">
13 -     <TD><INPUT TYPE ="reset">
14     </TABLE>
15   </FORM>
```

Dall'esecuzione dello script si otterrà come risultato il seguente modulo:

Il modulo invia al server i valori da sommare dopo averli inseriti nei due campi di input, chiamati rispettivamente numero1 e numero2.

I dati verranno inviati al clic del pulsante invia query, che attiverà il file di risposta, Somma.php, il cui codice è indicato di seguito:

```php
<?php
$n1=$_POST["numero1"];
$n2=$_POST["numero2"];
$somma=$n1+$n2;
echo "Risultato: $somma";
?>
```

Lo script produce la risposta che possiamo osservare nella schermata a lato, visualizzando nel browser la somma dei due valori inseriti nei due campi di input del Form.

Puoi trovare il codice di questo esempio nel file Somma.php.

La tecnica postback

In questo paragrafo esamineremo, attraverso un esempio, come leggere i dati provenienti da un Form con una connessione di tipo POST e sfruttando la tecnica postback, che consente alla pagina di inviare a se stessa il Form.
Fino ad adesso abbiamo visto come avviene il dialogo tra le pagine: la prima pagina statica contiene il tag <form> che viene inviato alla pagina php, la quale elabora i valori ricevuti.
La tecnica postback utilizza invece una sola pagina, di tipo dinamico, per effettuare entrambe le operazioni, cioè sia quella di raccolta dei dati dal Form che quella di elaborazione dei risultati. Sostanzialmente, la pagina invia informazioni a se stessa. Mediante il controllo sul contenuto di un campo possiamo verificare se la pagina è stata richiesta per la prima volta o se è stata richiamata in post da se stessa.

ESEMPIO

MODULO CON TECNICA POSTBACK
Utilizziamo la tecnica postback per fare in modo che form HTML e codice php di elaborazione convivano nella stessa pagina. Il codice dell'esempio è il seguente:

```php
<?php
//verifica se primo accesso attraverso variabile d'ambiente $_POST
//se contiene true significa che il modulo è stato inviato
//pertanto non si tratta del primo accesso alla pagina
if ($_POST)
{
    echo "il tuo nome: " . $_POST['username'];
    echo "<BR>";
    echo "la tua casella di posta: " .$_POST['email'];
    echo "<BR>";
}
else
{
    //utilizzo della variabile d'ambiente per reindirizzare
    //il form alla stessa pagina (tecnica post back)
    echo "<FORM ACTION= " . $_SERVER['PHP_SELF'] ." METHOD='POST'>";
    ?>
    <TABLE BORDER=0>
    <TR><TD>Nome utente:
    <TD><INPUT TYPE="text" NAME="username"><BR>
    <TR><TD> Email:
    <TD><INPUT TYPE="text" NAME="email"><BR>
    </TABLE><INPUT TYPE="submit" VALUE="invio dei dati">
    </FORM>
    <?php
}
?>
</BODY></HTML>
```

Si ottiene una prima pagina che mostra il modulo di immissione. Una volta riempito, viene spedito di nuovo alla stessa pagina, che si occupa di elaborarlo secondo la tecnica del postback.

Lo script inizia con la verifica della variabile d'ambiente $_POST, che contiene true quando la pagina riceve almeno un campo con il metodo POST. Se il Form avesse usato il metodo GET, sarebbe stato sufficiente modificare lo script utilizzando la variabile d'ambiente $_GET al posto di $_POST.

Il risultato dell'invio del modulo è riportato a lato.

Nello script possiamo notare come sia stata utilizzata anche la variabile d'ambiente $_SERVER['PHP_SELF'], che permette di conoscere l'indirizzo della pagina a cui reindirizzare il Form. Abbinata all'attributo ACTION, consente di reindirizzare la destinazione del Form alla pagina stessa.

Puoi trovare il codice di questo esempio nel file Postback.php.

Nell'esempio appena esaminato convivono nella stessa pagina sia il Form che le istruzioni che ne elaborano i dati ricevuti. In questo caso, l'attributo ACTION si riferisce alla stessa pagina contenente il Form.
Inoltre, sempre in questo esempio, abbiamo usato $_POST come metodo per verificare di aver ricevuto qualcosa dal Form, senza sapere tuttavia di aver ricevuto i dati corretti.
Per verificare invece che il singolo campo POST contenga qualcosa, utilizziamo la funzione isset(), come illustrato nell'esempio seguente.

ESEMPIO

UTILIZZO DELLA FUNZIONE ISSET()

L'esempio utilizza la funzione isset() per verificare che il campo ricevuto dal Form contenga qualcosa, cioè che esista effettivamente lo specifico campo. In questo esempio viene verificata la condizione che entrambi i campi contengano qualcosa attraverso una condizione logica and (&&).

```
1  <?php
2  //Form generato dinamicamente dalla pagina php
3  echo "<FORM METHOD='post' ACTION=".$_SERVER['PHP_SELF'].">
4  <TABLE WIDTH=30%>
5  <TR>
6  <TD>Primo Numero
7  <TD><INPUT TYPE='text' NAME='n1'>
8  <TR>
9  <TD>Secondo Numero
10 <TD><INPUT TYPE='text' NAME='n2'>
11 <CENTER>
12 <TR>
13 <TD><INPUT TYPE='reset' VALUE='Azzera'>
14 <TD><INPUT TYPE='submit' NAME='inviato' VALUE='Minore'>
15 </TABLE>
16 </FORM>";
17 //Verifica se i due sono stati settati
18 //Quindi contengono qualcosa
19 if(isset($_POST['n1'])&&isset($_POST['n2']))
20 {
21    //Verifica valore minore e assegnazione nella variabile $min
22    if($_POST['n1']<$_POST['n2'])
23       $min=$_POST['n1'];
24    else
25       $min=$_POST['n2'];
26    echo "Il numero minore è: $min";
27 }
28 else
29    echo "Inserisci i valori nel form";
30 ?>
```

L'esecuzione dello script fornisce il seguente risultato:

Puoi trovare il codice di questo esempio nel file Minore.php.

Lettura dei campi con foreach

Quando dobbiamo gestire un numero molto elevato di campi, possiamo ricorrere a un array associativo $_POST, che può essere visitato spostandosi all'interno dello stesso mediante un ciclo foreach.

Vediamo un esempio.

ESEMPIO

LETTURA CAMPI FORM MEDIANTE CICLO FOREACH

In questo esempio vogliamo accedere a tutti i campi POST ricevuti mediante un ciclo foreach. L'esempio si compone di due pagine, una pagina HTML contenente il Form e una seconda pagina php in grado di elaborarne i dati ricevuti.

La pagina HTML contenente il Form presenta il seguente codice:

```
6  <FORM NAME="modulo" METHOD="post" ACTION="foreach.php">
7  Nome e Cognome:<INPUT TYPE="text" NAME="username"><BR><BR>
8  Stato civile:<BR>
9  <INPUT TYPE="radio" NAME="civile" VALUE="coniugato">Coniugato<BR>
10 <INPUT TYPE="radio" NAME="civile" VALUE="non coniugato">Non coniugato<BR><BR>
11 Argomenti d'interesse:<BR>
12 <INPUT TYPE="checkbox" NAME="interesse">Tecnologia<BR>
13 <INPUT TYPE="checkbox" NAME="moto">Motori<BR>
14 <INPUT TYPE="checkbox" NAME="viaggi">viaggi<BR><BR>
15 Dove hai saputo del nostro sito?<SELECT NAME="sito">
16 <OPTION VALUE="giornali">Giornali</OPTION>
17 <OPTION VALUE="tv">TV</OPTION>
18 <OPTION VALUE="internet">Internet</OPTION>
19 <OPTION VALUE="amici">Amici</OPTION></SELECT><BR><BR>
20 <INPUT TYPE="reset" VALUE=" Azzera "><INPUT TYPE="submit" VALUE="  Invia  ">
21 </FORM>
```

Il codice della pagina php è il seguente:

```
1  <?php
2  $somma=0;
3  //ciclo che fa uso dell'array associativo $_POST
4  $ris=0;
5  // Ciclo per esaminare tutti gli elementi dell'array $_POST
6  foreach($_POST as $key => $value)
7  {
8    echo "Il campo POST: [".$key. "] contiene il valore= ".$value."<BR>";
9  }
10 ?>
```

Possiamo notare che nel ciclo foreach vengono utilizzate tutte le variabili $key e $value che contengono a ogni passaggio il nome e il contenuto dei campi POST ricevuti dal Form.

Eseguendo la prima pagina (HTML), appare il Form che deve essere compilato:

La pagina Foreach.php riceve i campi e mostra il valore contenuto in ciascuno di essi:

Puoi trovare il codice di questo esempio nei file Foreach.html e Foreach.php.

Il metodo GET e le query string

Finora abbiamo visto come inviare dati tramite i metodi GET e POST attraverso l'utilizzo di un Form.

In realtà, è possibile inviare le informazioni al server utilizzando direttamente la barra degli indirizzi del browser. Ovviamente, sul server deve essere memorizzato il file .php che elaborerà i dati inviati.

Con il metodo GET i dati vengono passati direttamente all'interno dell'indirizzo Web (URL) della pagina, il quale si presenterà accompagnato da un punto di domanda (?), seguito dai dati organizzati in coppie nome=valore, separati tra loro dal simbolo &.

Tali dati vengono chiamati query string e sono visibili a tutti nella barra dell'indirizzo del browser. L'esempio che segue mostra come utilizzare il metodo GET realizzando una semplice query string.

ESEMPIO

USO DELLE QUERY STRING

In questo esempio utilizziamo le conoscenze fin qui apprese per realizzare una semplice pagina che, ricorrendo alla tecnica postback, mostra al primo accesso una serie di link contenenti delle query string e, in base al clic effettuato dall'utente su uno di essi, invia la query string alla pagina stessa, che provvede alla elaborazione. I link sono stati creati in questo modo:

```
$me=$_SERVER['PHP_SELF'];
echo "<A HREF=".$me."?marca=Fiat&modello=Panda>Panda</A>
<A HREF=".$me."?marca=AlfaRomeo&modello=Giulietta>Alfa Giulietta</A>
<A HREF=".$me."?marca=Opel&modello=Panda>Opel Astra</A>
<A HREF=".$me."?marca=Citroen&modello=DS4>Citroen DS4</A>
<A HREF=".$me."?marca=Ford&modello=Ka>Ford Ka</A><BR>";
```

In tal modo, la pagina richiamata è quella contenuta nella variabile $me, cioè la pagina stessa, alla quale vengono inviati due campi (marca e modello) secondo il metodo GET.
Il codice della pagina è il seguente:

```
1  <?php
2  //variabile predefinita per l'indirizzo della pagina
3  $me=$_SERVER['PHP_SELF'];
4  //Verifica se la pagina è stata richiamata dal suo interno quindi se il metodo $_GET è true
5  if ($_GET)
6  {
7     //Stampa valore prelevato dall'array associativo $_GET
8    echo "Marca selezionata  : ".$_GET['marca']."<BR>";
9    echo "Modello selezionato: ".$_GET['modello']."<BR>";
10 }
11 else
12   //in questo caso si tratta del primo accesso quindi vengono mostrati a video i link
13   echo "<A HREF=".$me."?marca=Fiat&modello=Panda>Panda</A><BR>
14   <A HREF=".$me."?marca=AlfaRomeo&modello=Giulietta>Alfa Giulietta</A><BR>
15   <A HREF=".$me."?marca=Opel&modello=Panda>Opel Astra</A><BR>
16   <A HREF=".$me."?marca=Citroen&modello=DS4>Citroen DS4</A><BR>
17   <A HREF=".$me."?marca=Ford&modello=Ka>Ford Ka</A><BR>";
18 ?>
```

L'esecuzione dello script fornisce il seguente risultato:

Quando l'utente fa clic sul link, viene creato l'indirizzo seguente:

Puoi trovare il codice di questo esempio nel file Querystring.php.

METTITI ALLA PROVA

• Applicazione della tecnica postback • Lettura dei campi POST mediante ciclo foreach

Apri i file Foreach.html e Foreach.php

1 Modifica il codice dei due esempi fondendoli all'interno di un solo file php, in modo tale che i dati immessi dall'utente nel Form vengano elaborati nella stessa pagina.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Quale di queste affermazioni è corretta?
- **a** Le stringhe di query possono contenere spazi
- **b** Le stringhe di query possono contenere l'operatore "+"
- **c** Le stringhe di query possono contenere l'operatore "–"
- **d** Nessuna delle precedenti

2 La variabile $ACTION:
- **a** è una variabile superglobale
- **b** è una variabile definita dal programmatore
- **c** è inizializzata automaticamente con il nome della pagina corrente
- **d** contiene il valore dell'azione che si sta eseguendo

3 Per accodare due variabili in una stringa di query:
- **a** si utilizza l'operatore +
- **b** si utilizza l'operatore .
- **c** basta scriverle unite
- **d** si utilizza l'operatore &

4 La variabile $_SERVER[PHP_SELF] contiene:
- **a** la variabile superglobale SELF
- **b** il path completo della pagina che è attualmente in esecuzione
- **c** il nome della pagina in esecuzione
- **d** il path della home page del sito

5 L'istruzione:

```
echo "<A HREF=$SERVER['PHP_SELF']?scelta=1> Elenco dei dolci </A>";
```

- **a** è sintatticamente errata
- **b** è corretta e inizializza PHP_SELF al valore 1
- **c** è ricorsiva e richiama se stessa accodando scelta=1
- **d** è errata, in quanto indica un HREF di una pagina inesistente

6 L'istruzione:

```
echo "<A HREF=$SERVER['PHP_SELF']scelta=0&scelta1=Spaghetti>Spaghetti</A>";
```

- **a** è sintatticamente errata in quanto manca il ?
- **b** accoda una variabile alla stringa di query
- **c** accoda due variabili alla stringa di query
- **d** accoda tre variabili alla stringa di query

7 L'istruzione:

```
echo "<A HREF=\"$SERVER['PHP_SELF']?"."&a=scelta1\">Aggiorna ora </A>";
```

- **a** è sintatticamente errata
- **b** assegna al campo $a il valore contenuto in "scelta1"
- **c** assegna al campo &a il valore "scelta1"
- **d** assegna al campo a il valore "scelta1"

Verifica di fine Lezione

VERIFICA... le competenze

AREA DIGITALE

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Crea un Form che mostri due caselle di testo e un elenco (<SELECT>) in cui l'utente scriverà due numeri e sceglierà l'operazione matematica voluta (quattro operazioni ed elevamento a potenza). Usando la tecnica del postback restituisci il risultato a video.

2 Calcola e mostra la tabellina pitagorica di un numero inserito dall'utente mediante la tecnica postback.

3 Crea un Form che mostri 4 numeri interi casuali e un radio button che indica la base in cui trasformarli. Usa la tecnica postback per includere il codice necessario per mostrare il Form e i risultati della conversione.

4 Crea un Form che consenta di inserire un numero, quindi trasformalo invertendone le cifre usando la tecnica postback.

5 Crea un Form che mostri i piatti ordinabili in un ipotetico ristorante online. Il modulo contiene gli antipasti, i primi, i secondi, i contorni e le bevande. A seconda della scelta effettuata, deve mostrare il prezzo complessivo usando la tecnica del postback per leggere i campi del Form.

6 Crea un Form che legga una data (in formato gg/mm/aaaa), verifichi prima di tutto se la data è valida e quindi calcoli il giorno settimanale corrispondente sempre nella stessa pagina (postback).

7 Crea un Form come nella figura a lato.
La pagina deve ricevere i dati indicati e stamparli a video, ben incolonnati secondo la tecnica postback.

8 Crea un Form come nella figura a lato.
Se l'utente ha selezionato la voce Altro..., deve apparire un Form che consenta di selezionarla da un altro elenco come indicato nell'immagine qui sotto.

La pagina deve ricevere i dati indicati e stamparli a video ben incolonnati, usando la tecnica postback per includere tutto il codice in un'unica pagina php.

9 Crea un Form come nella figura seguente:

La pagina deve ricevere i dati indicati e stamparli a video ben incolonnati secondo la tecnica postback.

Stringhe e array

IN QUESTA LEZIONE IMPAREREMO...

- a gestire gli array associativi
- a utilizzare le stringhe
- ad applicare le funzioni predefinite per stringhe e array

MAPPA CONCETTUALE

didattica inclusiva

Gli array

I due tipi principali di array supportati da php sono:

– array numerici, dove ogni elemento viene individuato attraverso un indice numerico progressivo all'interno dell'array (il primo elemento del vettore è memorizzato nella posizione di indice 0);
– array associativi, dove ogni elemento viene indirizzato attraverso una chiave all'interno dell'array.

Possiamo dichiarare un array secondo due sintassi diverse:

– con il nome seguito dalle parentesi quadre a cui associare almeno un elemento che verrà inserito nella posizione indicata: per esempio, per inserire tale valore alla posizione zero scriviamo: $vet[]=0;
– attraverso la parola chiave array seguita dalle parentesi tonde che racchiudono gli elementi da assegnare all'array stesso, separati dalla virgola, per esempio: $vet=array(5,8,5,3);

Inoltre, possiamo inserire valori di tipo diverso anche all'interno dello stesso array:

```
// Tipo stringa alla posizione [0]
$vet[0] = "Riccardo";
// Tipo intero alla posizione [1]
$vet[1] = 65;
// Tipo float alla posizione [2]
$vet[2] = 13.4;
```

ESEMPIO

CARICAMENTO E STAMPA DI UN ARRAY

In questo esempio "carichiamo "un vettore con numeri casuali da 1 a 100 e poi lo stampiamo secondo due metodi, il primo che esegue un ciclo per tutti gli elementi e un secondo che sfrutta la funzione print_r() in grado di visualizzare il contenuto di un intero array:

```
1  <?php
2  //Definizione vettore
3  $vet[]=0;
4  //Ciclo di caricamento con numeri casuali
5  for($i=0;$i<20;$i++)
6     $vet[$i]=(int) rand(1,100);
7  echo "Stampa con ciclo<BR>";
8  //count è una funzione che restituisce la lunghezza del vettore
9  $c=count($vet);
10 echo  "Numero elementi presenti nel  vettore: $c <p>";
11 //visualizza tutti i valori contenuti nell'array con un ciclo
12 for($i=0;$i<$c;$i++)
13    echo  "$vet[$i]: ";
14 echo "<BR>";
15 //visualizza tutti i valori contenuti nell'array
16 //per fare questo al posto di echo usare print_r
17 echo "Stampa con print_r<BR>";
18 print_r($vet);
19 ?>
```

Il risultato dell'esecuzione dello script è il seguente:

```
127.0.0.1/Carica_array.php
App  Per un accesso rapido, inserisci i preferiti nella barra. Importa preferiti adesso...

Stampa con ciclo
Numero elementi presenti nel vettore: 20

17: 2: 49: 12: 61: 91: 72: 41: 10: 85: 85: 42: 71: 5: 72: 12: 5: 23: 68: 97:

Stampa con print_r
Array ( [0] => 17 [1] => 2 [2] => 49 [3] => 12 [4] => 61 [5] => 91 [6] => 72 [7] => 41 [8] => 10 [9] => 85 [10] =>
85 [11] => 42 [12] => 71 [13] => 5 [14] => 72 [15] => 12 [16] => 5 [17] => 23 [18] => 68 [19] => 97 )
```

Puoi trovare il codice di questo esempio nel file Carica_array.php.

Funzioni predefinite sugli array

php mette a disposizione delle funzioni predefinite, utili alla gestione degli array, che indichiamo di seguito:

```
array_search()            // restituisce la chiave del valore trovato oppure false se non esiste
count($vettore)           // restituisce il numero di elementi presenti nel vettore
min($vettore)             // restituisce il più piccolo elemento tra quelli presenti nel vettore
max($vettore)             // restituisce il più grande elemento tra quelli presenti nel vettore
sort($vettore)            // restituisce il vettore ordinato in modo crescente
rsort($vettore)           // restituisce il vettore ordinato in modo decrescente
shuffle($vettore)         // restituisce il vettore miscelato
reset($vettore)           // sposta il puntatore sul primo elemento dell'array
key($vettore)             // restituisce il valore della chiave dell'elemento corrente
current($vettore)         //r estituisce il contenuto della cella dell'elemento corrente
next($vettore)            // sposta il puntatore sull'elemento successivo del vettore
prev($vettore)            // sposta il puntatore sull'elemento precedente del vettore
```

Gli array associativi

Gli array associativi archiviano gli elementi insieme a valori chiave anziché secondo il tipico ordine lineare basato su di un indice numerico. Gli elementi di questi array (valori o value) non vengono identificati attraverso un numero, bensì tramite un contrassegno (chiave o key), grazie al quale è possibile associare gli elementi dell'array a termini univoci e quindi rendere più facile la ricerca di determinati elementi. La dichiarazione di un vettore associativo avviene tramite la parola chiave array():

```
$vet = array('gennaio'=>31,'febbraio'=>28,'marzo'=>31,'aprile'=>30);
```

Nell'esempio qui proposto, i mesi sono le chiavi (key) necessarie per accedere ai valori (value), rappresentati in questo caso dal numero dei giorni. Le chiavi e i valori vengono associati attraverso l'operatore di associazione =>.
Per effettuare un ciclo in cui leggere tutti gli elementi del nostro array possiamo usare la funzione each(), che estrae da ciascun elemento dell'array la chiave e il relativo valore. La funzione each() restituisce false quando l'array è terminato, pertanto può essere inserita in un ciclo while.

Questa funzione viene utilizzata insieme alla funzione list(), che colloca in due variabili la chiave e il valore fornito dalla funzione each( ):

```
list($nome,$valore) = each($vet)
```

Nel codice che segue le variabili $nome e $valore contengono rispettivamente la chiave e il valore dell'elemento corrente dell'array, inoltre il ciclo while termina quando each() restituisce false:

```
while( list($nome,$valore) = each($vet) )
{
 // stampa degli elementi del vettore
   echo $nome. " ".$valore;
}
```

ESEMPIO

CARICAMENTO E STAMPA DI UN ARRAY

In questo esempio dichiariamo un array associativo che contiene alcune capitali come chiave e le relative nazioni di appartenenza come valore.

Lo script, attraverso tre funzioni, visualizza le nazioni data la città, la città data la nazione e l'elenco completo delle nazioni. La prima funzione (nazione_per_citta), utilizzando la città come chiave, ottiene automaticamente la nazione passandola al vettore come elemento tra parentesi quadra.

La seconda funzione (citta_per_nazione) visualizza invece la città data la nazione: per fare questo effettua una ricerca nell'array mediante la funzione array_search(), che riceve il valore (in questo caso, la nazione) e restituisce la chiave relativa, cioè la città. In entrambe le funzioni, prima di effettuare la ricerca, viene verificata l'esistenza dell'elemento da cercare: nella prima funzione attraverso la funzione isset(), che restituisce true se il vettore passato come parametro esiste, mentre nella seconda funzione la stampa viene effettuata solo se array_search() restituisce true.

La terza funzione (stampa_tutto()), infine, effettua una scansione del vettore per visualizzare tutti gli elementi dell'array.

```
1  <?php
2  function nazione_per_citta($vet,$citta)
3  {
4      //se la chiave esiste stampa la città passata come parametro e il relativo valore
5      if(isset($vet[$citta]))
6      {
7          echo "la città: ".$citta. " è in: " . $vet[$citta];
8          echo "<HR>";
9      }
10 }
11 function citta_per_nazione($vet,$nazione)
12 {
13     //Uso la funzione array_search per cercare la chiave dato il valore
14     if ($a=array_search($nazione,$vet))
15         echo "la capitale della: ".$nazione. " è in: " . $a;
16     else
17         echo "nazione non trovata";
18     echo "<HR>";
19 }
20 function stampa_tutto($vet)
21 {
22     //posizionamento all'inizio del vettore
23     reset($vet);
24     //estrazione del primo elemento dell'array
25     $tmp=current($vet);
26     //se elemento presente avviene la stampa
27     if($tmp)
28         {
29         echo "$tmp<BR>";
30         while($tmp = next($vet))
31             echo "$tmp<BR>";
32         }
33     else
34         echo "array vuoto";
35     echo "<HR>";
36 }
37 //definizione array associativo con stile chiave=>valore
38 $info=array('Madrid'=>'Spagna','Roma'=>'Italia','Londra'=>'Regno Unito','Berlino'=>'Germania','Parigi'=>'Francia');
39 //stampa del terzo elemento
40 citta_per_nazione($info,'Regno Unito');
41 nazione_per_citta($info,'Roma');
42 stampa_tutto($info);
43 ?>
```

```
la capitale della: Regno Unito è in: Londra
---
la città: Roma è in: Italia
---
Spagna
Italia
Regno Unito
Germania
Francia
---
```

L'esecuzione dello script produce il risultato riportato a lato.

Puoi trovare il codice di questo esempio nel file Array_associativo.php.

Le stringhe

Le stringhe in php sono spesso utilizzate per gestire i dati provenienti dai campi Form, dai file o dai database. Alcune funzioni predefinite che le gestiscono sono le seguenti:

```
strlen($stringa)                              // restituisce la lunghezza della stringa
strtolower($stringa)                          // restituisce la frase in caratteri tutti minuscoli
strtoupper($stringa)                          // restituisce la frase in caratteri tutti maiuscoli
strrev($stringa)                              // restituisce la frase con i caratteri di ogni parola invertiti
str_replace("Frase","Stringa",$stringa)       // restituisce la frase di partenza con "Frase" al posto di "Stringa"
```

ESEMPIO

LE FUNZIONI STRINGA

Il codice di questo esempio è molto semplice: viene elaborata la stringa presente nella variabile $stringa per estrarne alcune informazioni:

```
<?php
$stringa = "Salve, sono una Stringa";
//restituisce la lunghezza della stringa (23)
echo strlen($stringa);
echo "<BR>";
//restituisce la frase in caratteri tutti minuscoli
echo strtolower($stringa);
echo "<BR>";
//restituisce la frase in caratteri tutti maiuscoli
echo strtoupper($stringa);
echo "<BR>";
//restituisce la frase con solo il primo  carattere in maiuscolo
echo ucfirst($stringa);
echo "<BR>";
//restituisce la frase con il primo  carattere di ogni parola in maiuscolo
echo ucwords($stringa);
echo "<BR>";
//restituisce la frase con i caratteri di ogni parola invertiti.
echo strrev($stringa);
echo "<BR>";
//restituisce la frase di partenza con "Frase" al posto di "Stringa"
echo str_replace("Stringa","Frase",$stringa);
?>
```

Eseguendo lo script otteniamo il seguente output:

```
23
salve, sono una stringa
SALVE, SONO UNA STRINGA
Salve, sono una Stringa
Salve, Sono Una Stringa
agnirtS anu onos ,evlaS
Salve, sono una Frase
```

Puoi trovare il codice di questo esempio nel file Funzioni_stringa.php.

METTITI ALLA PROVA

➜ • Utilizzo del postback • Utilizzo degli array

Apri il file Poker.php

1 Modifica il codice dell'esempio in modo tale che l'utente possa selezionare la carta da cambiare scrivendola in un campo di un Form. La carta modificata viene passata alla pagina stessa in postback e il codice deve provvedere alla stampa del risultato.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Barra l'affermazione corretta in base al seguente segmento di codice:

```
for ($xx=0; $xx++ ; $xx<12)
{ $mesi1[ ]=xx; }
```

a inizializza correttamente un array di 12 elementi
b definisce e riempie correttamente un vettore di 12 elementi
c produce un errore se l'array non è precedentemente dichiarato
d produce un errore in quanto non sono indicate le posizioni in cui si inseriscono gli elementi

2 Indica quale definizione di array è corretta:
a &animali[]=0
b %animali[]=0
c $animali[]=0
d (animali())=0

3 La seguente istruzione:

```
$animali[ ]="gatto";
```

a inserisce un elemento nella posizione 0
b è errata perché incompleta
c inserisce un elemento nell'ultima posizione di un array associativo
d inserisce un elemento nel primo posto libero in un array associativo

4 Il codice:

```
$riga = each($animali);
echo "Cella[0]  ".$riga[0];
echo "  Cella [1]  ".$riga[1];
```

a è errato
b non restituisce nulla
c visualizza il contenuto del vettore ($animali)
d visualizza le prime due posizioni del vettore ($animali)

5 L'istruzione:

```
$animali[3].=" cammello";
```

a è errata
b inserisce un elemento nella posizione 3
c inserisce un elemento nella posizione con indice "3"
d concatena il contenuto della posizione 3

6 Il codice:

```
$animali[1]="gatto";
$animali[3]="topo";
```

a inizializza correttamente un array di 4 elementi
b inizializza correttamente un array di 3 elementi
c inizializza correttamente un array di 2 elementi
d inizializza erratamente un array

7 Il codice:

```
reset($animali);
while (list($chiave,$valore) = each($animali))
{
 echo chiave." ".$valore."</b><br>";
}
```

a è errato
b visualizza correttamente il contenuto dell'array ($animali)
c visualizza il contenuto del vettore ($animali) con valore della chiave $chiave desiderata
d visualizza gli elementi uguali del vettore ($animali)

VERIFICA... le conoscenze

8 Il codice:

```
sort($animali);
for ( $xx=0; $xx<count($animali) ; $xx++ )
{
 echo "<br>Pos=> ".$xx." Chiave ";
 echo key($animali)." => Valore <b>".current($animali)."</b>";
 next($animali);
}
```

a è errato
b visualizza il contenuto ordinato secondo la chiave
c visualizza il contenuto ordinato secondo il valore
d visualizza il contenuto ordinato secondo entrambi

9 Il Il codice:

```
$zoo=array (
"gatto"=>array("nome"=>"pippo","colore"=>"rosso", "eta"=>"3 anni","cibo"=>"carne"),
"micio"=>array("nome"=>"fufi","colore" =>"bianco","eta"=>"1 anni","cibo"=>"latte"));
```

a è errato
b crea un array bidimensionale
c crea un array tridimensionale
d inserisce due elementi in un array zoo

10 Il codice:

```
reset($animali);
while (list($chiave,$valore) = each($animali))
{
 echo chiave." ".$valore."</b><br>";
}
```

a è errato
b visualizza correttamente il contenuto dell'array ($animali)
c visualizza il contenuto del vettore ($animali) con valore della chiave $chiave desiderata
d visualizza gli elementi uguali del vettore ($animali)

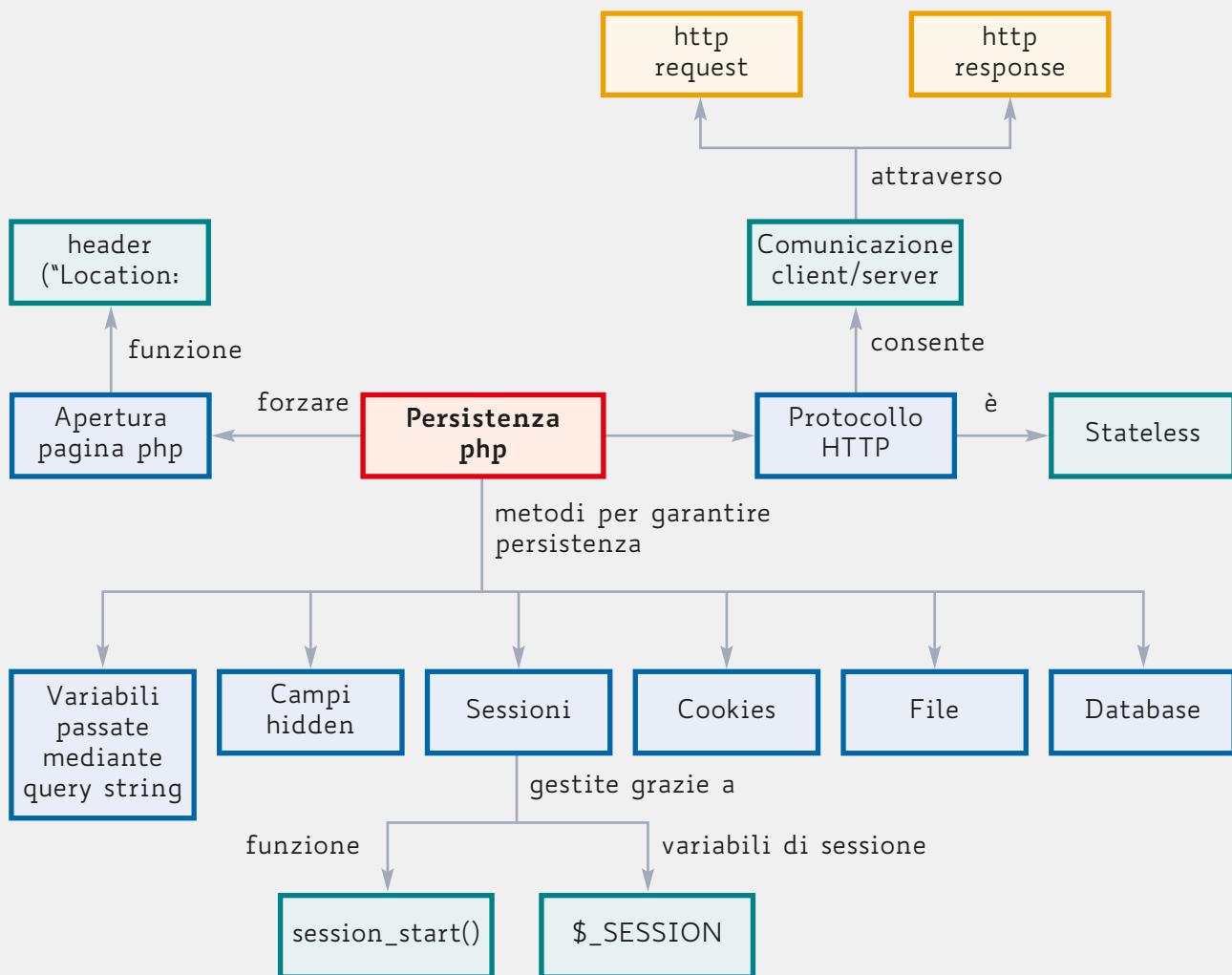
VERIFICA... le competenze

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Dopo aver ricevuto un array di interi stampa a video tutti i numeri contenuti nell'array, quindi stampa il minimo e il massimo dei numeri contenuti nell'array e la media aritmetica.

2 Riempi in modo casuale un array sequenziale di 20 numeri interi, quindi mostra a video quanti sono i pari e quanti i dispari.

3 Dopo aver ricevuto un elenco di nomi da una casella di testo multiriga (<TEXTBOX>), inviali alla pagina stessa che deve trasformarli in una stringa da stampare a video.

4 Crea un array associativo provincia-sigla della tua Regione e mostralo in un Form.

5 Crea un array associativo cognome-nome degli alunni di una classe e mostralo in un Form. Accanto a ciascun nome deve apparire un check button. Mediante la tecnica postback mostra i nomi selezionati.

6 Crea un array associativo cognome-nome-numero_maglia dei calciatori della tua squadra e mostrali in un Form in ordine in ordine di numero maglia o di cognome, in base a quanto deciso dall'utente.

7 Crea un array multi-dimensionale contenente i dati anagrafici di un insieme di clienti e visualizzali sotto forma di tabella.

8 Dato un array di stringhe libri, trasformale in un'unica stringa separandoli con il tag <BR> e mostrali a video.

9 Dopo aver ricevuto un elenco di frutti da un elenco MULTIPLE, trasferisci i valori contenuti nelle variabili in un array e stampa sia i valori dei campi che l'array.

10 Crea un Form che riceva dall'utente un elenco di persone e le trasferisca in un array associativo in cui le chiavi sono i cognomi e i valori sono i nomi. La pagina deve creare un insieme di variabili leggendole dall'array.

11 Leggi da un Form 10 nazioni e le relative capitali e trasferiscile in un array associativo, quindi stampalo ordinato per nazione.

12 Leggi da un Form 10 animali e il rispettivo verso e trasferiscili in un array associativo, quindi stampalo ordinato alfabeticamente.

13 Definisci un array multi-dimensionale contenente i dati anagrafici di un insieme di clienti e visualizzali sotto forma di tabella.

14 Crea un Form che riceva una stringa in input e stampi:
- il numero di consonanti e vocali che contiene;
- il numero di caratteri uguali a quanto inserito dall'utente;
- il numero di caratteri di tipo numerico;
- la frequenza con cui ogni carattere appare nella stringa;
- il numero di parole (ogni parola termina con un carattere di punteggiatura o con lo spazio).

5 La persistenza nel dialogo http

IN QUESTA LEZIONE IMPAREREMO...

- il concetto di persistenza tra pagine php
- a realizzare la persistenza con campi hidden, cookies e sessioni

MAPPA CONCETTUALE

http request
http response
attraverso
header ("Location:
Comunicazione client/server
funzione
consente
Apertura pagina php
forzare
Persistenza php
Protocollo HTTP
è
Stateless
metodi per garantire persistenza
Variabili passate mediante query string
Campi hidden
Sessioni
Cookies
File
Database
gestite grazie a
funzione
variabili di sessione
session_start()
$_SESSION

La persistenza in php

Il protocollo per il Web (HTTP) è di tipo stateless, si tratta cioè di un protocollo senza memoria di stato, in cui ciascuna coppia request_client/response_server è indipendente dalle altre. Sin dall'inizio dell'espansione del Web, questo si è rivelato un forte limite. Per ovviare a questo inconveniente, esistono delle tecniche che simulano lo stato in una tipica connessione client/server.
Le tecniche che consentono di memorizzare informazioni sul server possono essere suddivise in metodologie a carattere temporaneo (utili per esempio al trasferimento temporaneo tra pagina e pagina, oppure durante una sola sessione di lavoro) e definitivo (utili per esempio al salvataggio di informazioni per un tempo indefinito).

Appartengono alla prima categoria le tecniche di seguito elencate.
– Uso di variabili passate alla pagina successiva, aggiungendole alla query string:

```
http://www.sito.it/pagina.php?visite=1
...
http://www.sito.it/pagina.php?visite=15
```

– Uso dei campi nascosti (cioè di tipo hidden):

```
<FORM METHOD=”post” ACTION=” http://www.sito.it/pagina.php “>
...
<INPUT TYPE=”hidden” NAME=”visite” VALUE=”1”>
...
</FORM>
...
<FORM METHOD=”post” ACTION=” http://www.sito.it/pagina.php “>
...
<INPUT TYPE=”hidden” NAME=”visite” VALUE=”2”>
...
</FORM>
```

– Uso dei cookies. Nel caso di ricorso a questa tecnica, il server, oltre alla pagina richiesta, può includere nell'header della risposta la richiesta al client di ricevere un cookie. Il cookie è di fatto una variabile che ha un nome e un valore. Se abilitato a ricevere cookies, il client salva il nome e il valore della variabile sul proprio disco in un file che ha lo stesso nome del server che ha inviato il cookie. Il cookie scade dopo un certo tempo specificato dal server. I cookies, tuttavia, presentano una debolezza enorme: le informazioni che si vogliono conservare tra una pagina e l'altra vengono salvate in un file sul disco del client, e sono pertanto modificabili dall'utente.
– Uso delle sessioni.

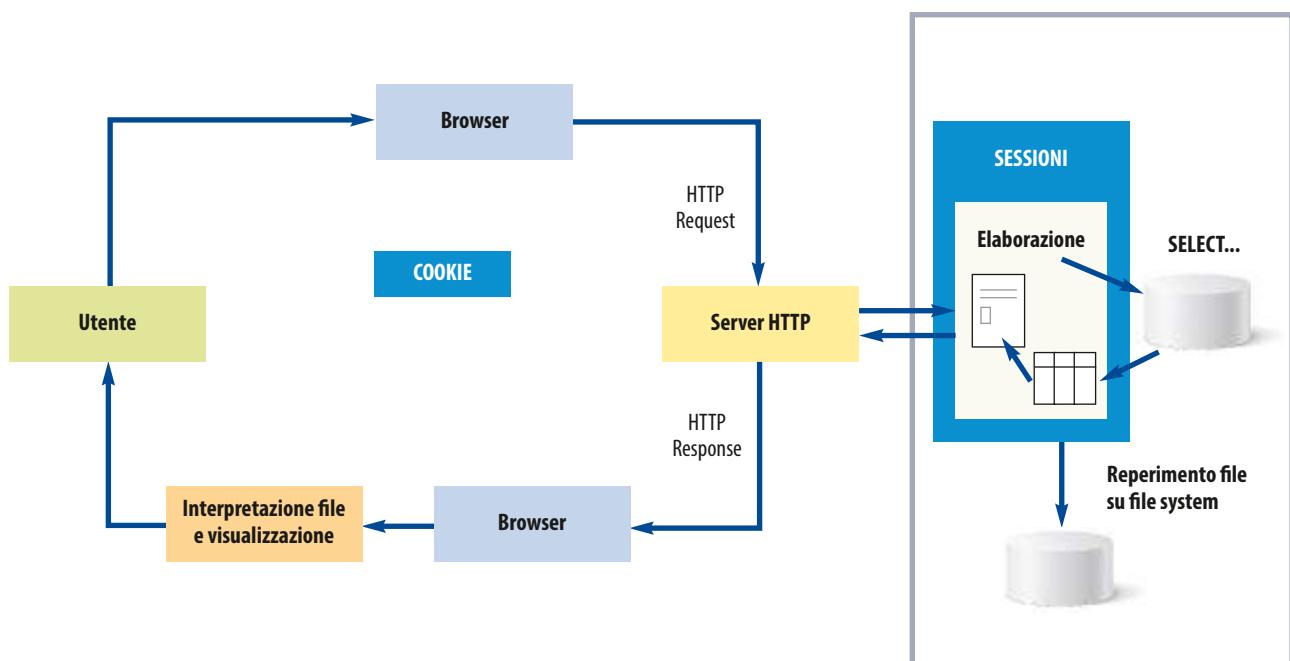
Fanno invece parte della seconda categoria le tecniche di seguito indicate.
– Uso del file system. In questo caso è il server a memorizzare man mano le informazioni all'interno di file memorizzati nel proprio sistema, come per esempio sull'hard disk. Questo metodo viene utilizzato solo per memorizzare informazioni che non sono oggetto di continue interrogazioni o elaborazioni, situazioni per le quali risulta preferibile l'utilizzo di un database.
– Uso dei database. I database rappresentano la tecnica privilegiata per realizzare l'architettura Three-Tier, tipica della conversazione client/server con linguaggi server side. In questo caso, la quantità di dati da gestire deve giustificarne l'uso: non avrebbe infatti senso utilizzare un data-

base per memorizzare solo informazioni temporanee tra pagine, mentre è fondamentale nel caso di informazioni massive e permanenti.

Esaminiamo più in dettaglio, nei paragrafi che seguono, la tecnica delle sessioni e dei cookies.

Le sessioni

Una sessione http è un insieme di request e response tra client e server che condividono uno stato, rappresentato da informazioni che persistono tra una connessione e le successive.

Le sessioni http vengono attivate, in php, dalla funzione session_start( ), che deve essere scritta in testa alla pagina, prima di qualunque altra riga di codice HTML:

```
<?php
session_start( );
?>
<HTML>
...
```

La funzione session_start() memorizza in un cookie (PHPSESSID) sul computer del client un identificativo di sessione (Session ID) casuale, l'unica informazione memorizzata sul client.

La tecnica delle sessioni ricorre contemporaneamente all'uso di tre strategie:

1 viene utilizzato un valore di codice identificativo (PID) molto difficile da indovinare (per esempio: "H6K7DHDIE737462BSTG37");
2 i dati sensibili vengono memorizzati sul disco del server, associandoli al codice identificativo dell'utente SID (Session IDentifier);
3 il cookie che contiene il codice identificativo viene impostato a durata temporale 0, inoltre viene eliminato chiudendo il browser.

L'esempio che segue mostra come usare le sessioni per calcolare le visite dello stesso utente a una pagina.

ESEMPIO

CONTA VISITE DELLO STESSO UTENTE

Lo script indicato di seguito esegue il calcolo delle visite effettuate alla pagina dallo stesso utente, utilizzando le sessioni per memorizzare il valore del conteggio. Se proviamo a chiudere e a riaprire il browser, il contatore di visite verrà azzerato, in quanto alla chiusura del browser vengono eliminate tutte le sessioni aperte. Lo script inizia con la funzione session_start(), che crea nel cookie una variabile chiamata PHPSESSID per attivare le sessioni.

```
1  <?php
2  //attivazione delle sessioni
3  //obbligatorio come prima riga in PHP ogni volta che si usano le sessions
4  session_start();
5  //se la variabile di sessione non esiste
6  if(!isset($_SESSION['numero_visite']))
7  {
8      //settaggio variabile sessione a 1
9      $_SESSION['numero_visite']=1;
10 }
11 else
12 {
13     //se esiste già incremento variabile sessione
14     $_SESSION['numero_visite']++;
15 }
16 ?>
17 <HTML>
18 <BODY>
19 Il numero di volte che ti sei connesso a questa pagina è
20 <B>
21 <?php
22 //visualizzazione variabile di sessione
23 echo $_SESSION['numero_visite'];
24 ?>
25 </B>
26 </BODY>
27 </HTML>
```

Ovviamente, affinché lo script funzioni è necessario avere i cookies attivati sul browser.

Come possiamo notare, premendo più volte il tasto F5 otteniamo:

Il numero di volte che ti sei connesso a questa pagina è **18**

Puoi trovare il codice di questo esempio nel file Contavisite_utente.php.

Vediamo adesso un altro esempio, che mostra come usare le sessioni per effettuare l'autenticazione di un utente.

ESEMPIO

AUTENTICAZIONE "STUPIDA" DI UN UTENTE

In questo esempio viene effettuata una autenticazione definibile come "stupida", in quanto il nome dell'utente autorizzato è presente in una variabile all'interno del codice stesso. Un sistema più valido prevede invece che la lettura avvenga da un file, oppure da un database che contiene una tabella con i dati degli utenti registrati.

Si compone di due pagine, la prima delle quali consente di effettuare il login (Accedi_dummy.php), richiede i dati all'utente da autenticare, quindi, tramite la tecnica postback, richiama la pagina stessa passando come parametri i campi nome utente e password. Se i campi vengono riconosciuti, viene richiamata la pagina Accedi_dummy_aut.php.

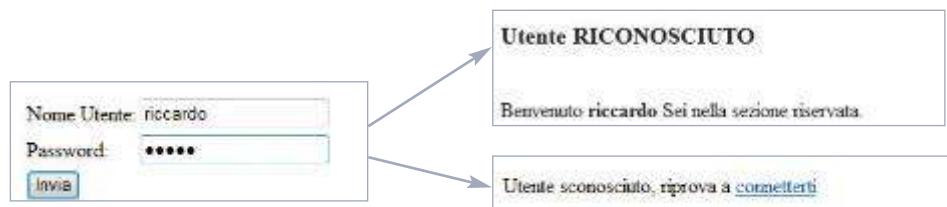
La prima parte del codice della pagina Accedi_dummy.php effettua il controllo sul settaggio della variabile $_POST['utente'], in grado di verificare se si tratta del primo accesso o del successivo. Nel caso di primo accesso viene mostrato il Form di login.

```
1   <?php
2   //inizializzazione sessioni
3   session_start();
4   //Tecnica postback: verifica se non è il primo accesso alla pagina
5   if (!isset($_POST['utente']))
6   {
7       //se primo accesso visualizzazione form che richiama la pagina stessa
8       //usando la variabile d'ambiente $PHP_SELF per evitare di riscrivere il nome della pagina
9   ?>
10  <HTML><HEAD><TITLE>Login</TITLE></HEAD>
11  <BODY>
12  <FORM METHOD="post" ACTION="<?php echo $_SERVER['PHP_SELF'] ?>">
13  Nome Utente:<INPUT NAME="utente"><BR>
14  Password:<INPUT TYPE="password" NAME="password"><BR>
15  <INPUT TYPE="submit" VALUE="Invia">
16  </FORM></BODY></HTML>
17  <?php
18  }
```

Header

Come possiamo notare alle righe 42, 50 e 59, viene usata la funzione header("Location: pagina.php") per reindirizzare alla pagina php indicata. Tale funzione, tuttavia, non può essere preceduta da istruzioni di visualizzazione (echo), codice HTML e di inclusione di codice esterno (include).

La seconda parte del codice viene eseguita quando la variabile è stata settata, quindi quando l'utente ha riempito i campi del Form di login. Se l'utente è riconosciuto come autorizzato, attraverso il confronto con l'array associativo $utenti[] (righe 33-36), viene settata la variabile di sessione $_SESSION['utente'] (riga 40), quindi chiamata la pagina Accedi_dummy_aut.php (riga 42); in caso contrario, vengono cancellate le variabili di sessione con la funzione unset (righe 47-48) e richiamata la pagina stessa (riga 50) con la funzione **header**. La stessa operazione viene effettuata per confrontare la password (righe 56-59).

```
18  }
19  else
20  {
21      //se secondo accesso alla pagina avviene controllo autorizzazioni
22      //la variabile user_id contiene il nome utente in minuscolo
23      $user=strtolower($_POST['utente']);
24      ///la variabile pwd contiene la password
25      $pwd=$_POST['password'];
26      //l'array utenti contiene i nomi degli utenti ammessi e le relative password
27      $utenti = array(
28        'admin' => 'qwerty',
29        'paolo' => '1234',
30        'riccardo' => 'pippo'
31      );
32      //verifica se l'array contiene contiene un nome utente come inserito nel form
33      if (isset($utenti["$user"]))
34         //in caso positivo verifica se l'array contiene una password come inserita nel form
35      {
36         if ($utenti["$user"] == $pwd)
37         {
38            //in caso positivo avviene la creazione della variabile di sessione
39            //con il nome dell'utente
40            $_SESSION['utente']=$_POST['utente'];
41            //quindi viene richiamata la pagina autorizzato.php
42            header("Location: Accedi_dummy_aut.php");
43         }
44         else
45         {
46            //se password non riconosciuta vengono eliminate le variabili ricevute dal form
47            unset($_POST['utente']);
48            unset($_POST['password']);
49            //viene richiamata di nuovo la pagina di login
50            header("Location: ".$_SERVER['PHP_SELF']);
51         }
52      }
53      else
54      {
55         //se il nome utente non è riconosciuto vengono eliminate le variabili ricevute dal form
56         unset($_POST['login']);
57         unset($_POST['password']);
58         //viene richiamata di nuovo la pagina di login
59         header("Location: ".$_SERVER['PHP_SELF']);
60      }
61  }
62  ?>
```

Il codice della seconda pagina Accedi_dummy_aut.php che riceve i dati di login dalla prima pagina effettua, in primo luogo, il controllo sul settaggio della variabile di sessione (riga 10). In caso positivo,

apre la sezione per l'utente autorizzato, in caso contrario appaiono un messaggio e un link per richiamare la pagina principale (riga 25):

```
1  <?php
2  //inizializzazione sessioni
3  session_start();
4  ?>
5  <HTML><BODY>
6  <?php
7  //verifica se variabile di sessione è attiva
8  //questo per evitare che se un utente conosce la pagina
9  //la possa aprire direttamente senza passare da Accedi_dummy.php
10 if (isset($_SESSION['utente']))
11 {
12 ?>
13     <H3>Utente RICONOSCIUTO</H3><BR>
14 <?php
15     //visualizzazione nome utente
16     echo "Benvenuto<B> " . $_SESSION['utente'];
17 ?>
18     </B>Sei nella sezione riservata.
19 <?php
20 }
21 else
22 {
23     //utente non riconosciuto in quanto la variabile di sessione non è stata settata
24 ?>
25     Utente sconosciuto, riprova a <A HREF="Accedi_dummy.php">connetterti</A>
26 <?php
27 }
28 ?>
29 </BODY></HTML>
```

Dall'esecuzione dello script si ottiene che, se l'utente è registrato, appare, nella schermata di seguito riprodotta, la pagina indicata a destra in alto, altrimenti, si ottiene come risultato un messaggio di errore con un link ipertestuale per tornare alla pagina di login.

Puoi trovare il codice di questo esempio nei file Accedi_dummy_aut.php e Accedi_dummy.php.

I cookies

$_COOKIE è un array associativo che contiene le variabili inviate allo script corrente attraverso cookies. I cookies non sono altro che piccoli file memorizzati sul client su richiesta esplicita del server, dei contenitori di poche informazioni che il server invita il client a rimandargli ogni volta che invia una richiesta. Da queste informazioni il server può riconoscere il client univocamente. Tuttavia, dobbiamo sempre tenere presente che il client si può rifiutare di accettare o di creare i cookies.
I cookies vengono utilizzati come soluzione alla mancanza di stato nelle connessioni http.

L'esempio che segue mostra come utilizzare i cookies per tenere conto delle visite dello stesso utente.

ESEMPIO

CONTA VISITE CON I COOKIES

Vogliamo realizzare un semplice script che mostri a video il numero di visite effettuate. Per fare questo, lo script verifica se il cookie visite è stato creato (riga 3). In caso positivo, viene incrementato

il contenuto del cookie chiamato visite nella variabile $numero_visite. Se invece alla riga 3 viene dato esito negativo, la variabile $numero_visite viene inizializzata a 1.

Anche in questo caso, siamo di fronte a uno script classificabile come "stupido": il conteggio infatti è relativo allo stesso utente, in quanto il cookie viene salvato sul client; inoltre, alla chiusura del browser il conteggio ripartirà da 1. Per effettuare un conteggio "serio" avremo bisogno dei file, che saranno oggetto della prossima lezione (Lezione 6 online, "I file e l'upload in php).

Il numero delle visite viene salvato nel cookie (riga 16) mediante la funzione setcookie, che riceve tre parametri: il nome del cookie (visite), la variabile da scrivere ($numero_visite) e il tempo di "vita" del cookie, in questo caso 3600 secondi (1 ora). Infine, viene visualizzato il totale.

```
1  <?php
2  // verifica se il cookie 'visite' esiste
3  if(isset($_COOKIE['visite']))
4  {
5     //in caso positivo incremento del cookie visite di 1
6     //viene eseguito il casting all'intero in quanto il cookie è sempre di tipo stringa
7     $numero_visite=(int) $_COOKIE['visite']+1;
8  }
9  else
10 {
11    //se è la prima visita viene creato il cookie e assegnato ad esso 1
12    //tramite la variabile numero_visite
13    //il cookie creato ha un tempo di durata di 3600 secondi = 1 ora
14    $numero_visite=1;
15 }
16 setcookie('visite', $numero_visite, time()+3600);
17 if( $numero_visite==1 )
18 {
19    echo "<H2>Benvenuto!</H2>";
20    echo "Nell'ultima ora hai visitato questa pagina 1 volta";
21 }
22 else
23 {
24    echo "<H2>Bentornato!</H2>";
25    echo "Nell'ultima ora hai visitato questa pagina $numero_visite volte";
26 }
27 ?>
```

Nel nostro esempio, abbiamo eseguito due processi del browser per simulare la connessione da parte di diversi utenti: come possiamo notare, ciascun browser contribuisce a incrementare le presenze. In realtà, abbiamo barato un po', in quanto, se ci fossimo collegati allo stesso script da due host diversi, ci sarebbero stati due conteggi separati poiché, come abbiamo già sottolineato, i cookie vengono salvati sul client e non sul server.

Puoi trovare il codice di questo esempio nel file Visite_cookie.php.

METTITI ALLA PROVA

➜ • Utilizzo della persistenza con i cookies

Apri il file Cookie.php

1. Aggiungi una funzione che mostri il contenuto dei cookies memorizzati, cioè delle scelte musicali effettuate.
2. Modifica il codice dello script in modo tale che l'utente possa anche decidere di eliminare i cookies che memorizzano le scelte effettuate.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Quale tra le seguenti affermazioni riguardanti la funzione unset è vera?
a unset() cancella un campo del Form
b unset() equivale a isset()
c unset() cancella una variabile di sessione
d unset() equivale a empty()

2 Il codice:

```
echo "Salve a tutti<BR>";
header("Location: pagina2.php");
```

a è errato in quanto prima di header vi è una echo
b apre il file pagina2.php
c chiude il file pagina2.php
d è errato in quanto location deve essere scritto in minuscolo

3 Gli array associativi $_GET[] e $_POST[]:
a devono essere definiti dall'utente
b devono essere inizializzati
c sono variabili superglobali
d sono servizi HTTP

4 Il SID di una sessione:
a è sempre un numero
b può essere definito dall'utente
c dipende dal sistema operativo
d è sempre alfabetico

5 L'istruzione:

```
setcookie("nome","pippo", time()+30);
```

a è incompleta, quindi dà un messaggio di errore
b setta la variabile cookie con validità 30 giorni
c setta la variabile cookie con validità 30 minuti
d setta la variabile cookie con validità 30 secondi

6 L'istruzione:

```
$_SESSION['nome']=$_POST["nome"];
```

a è sintatticamente errata
b non fa nulla
c trasferisce al campo POST la variabile di sessione letta
d trasferisce alla variabile di sessione il campo POST letto

7 L'istruzione:

```
echo "<INPUT TYPE=hidden NAME=citta VALUE=".$citta.">";
```

a nasconde la variabile se il valore è uguale alla variabile $citta
b genera una casella di input
c inizializza un campo nascosto
d cancella un campo nascosto

8 L'istruzione:

```
echo "<A HREF=dolci.php?scelta=1&nome=Paolo>1 - Elenco dei dolci </A>";
```

a è sintatticamente errata
b genera una query string con due campi nascosti
c genera una query string e inizializza due campi
d genera una query string errata

VERIFICA... le competenze

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Mostra un Form composto da due list box, regione e provincia di nascita, quindi trasferiscili in una seconda pagina mostrandoli a video.

2 Memorizza, mediante i cookies, il nome e il cognome di un visitatore letto da un Form.

3 Definisci una sessione con tre variabili che ricordino il colore dello sfondo, il tipo di font e la dimensione dei caratteri che un utente seleziona in un Form.

4 Mostra un Form che consenta all'utente di selezionare un insieme di 10 domande vero o falso e ne riporti in una successiva pagina le correzioni con la valutazione.

5 Mostra un Form dove l'utente può selezionare la squadra preferita, memorizzandola in una sessione; il sito deve poi ricordarla anche nelle pagine successive, riservate solo a chi conosce la password.

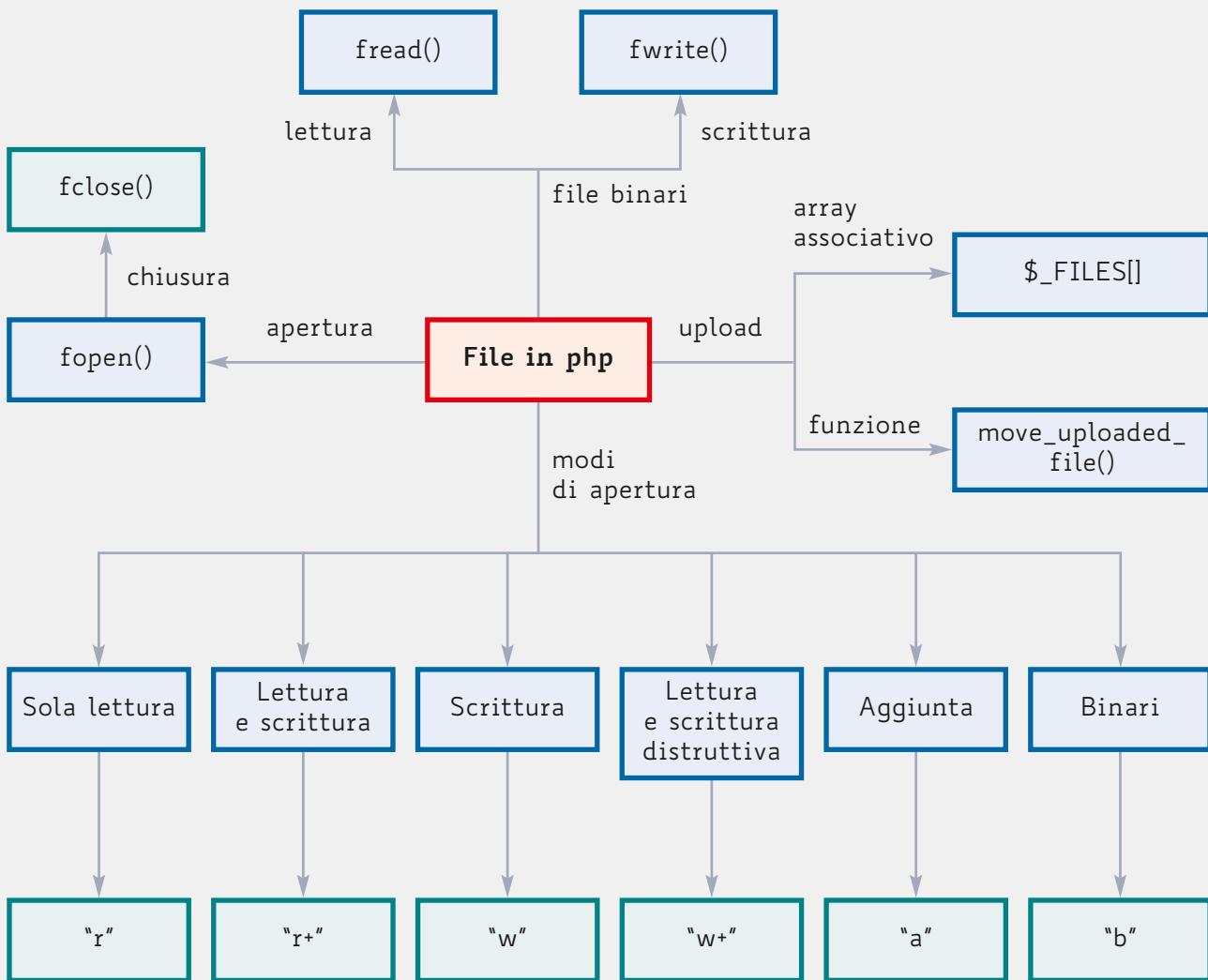
6 Mostra un Form dove l'utente può entrare in una sezione riservata in cui selezionare due generi musicali che verranno conservati per 2 giorni.

7 Crea una pagina che simuli un ristorante online. L'utente seleziona il piatto della portata da un elenco a discesa (<SELECT>), quindi il piatto prescelto verrà conservato e passato alla sezione successiva, fino ad arrivare al conto, utilizzando le sessioni. Utilizza almeno 6 portate:
antipasti – primi – secondi – contorni – dessert – formaggi – frutta – caffè o amari

8 Mostra un Form di registrazione, con dati anagrafici, hobbies e sport riportando in una successiva pagina riepilogativa tutte le scelte effettuate dall'utente.

9 Mostra un Form che permetta all'utente di selezionare delle preferenze (colori sfondo, segno zodiacale ecc.), in modo che ai successivi accessi (per una settimana) gli venga proposta una home page personalizzata.

10 Per la prestazione lavorativa di un impiegato ogni giorno vengono registrate l'ora di ingresso e di uscita dall'ufficio. Scrivi una pagina per inserire i dati dell'impiegato e calcolare il totale delle ore e dei minuti lavorati, visualizzando la paga giornaliera facendo inserire la paga oraria.

11 Mostra un Form in cui simulare un'asta online. In ciascuna pagina appare un quadro, che l'utente può scegliere se acquistare o meno scrivendone il prezzo. Al termine dell'asta, devono apparire il conto e i quadri aggiudicati.

12 Crea un sito in cui mostrare gli accessi di ogni singola pagina di cui è formato. Il sito deve possedere almeno 10 pagine.

13 Mostra un Form che consenta all'utente di selezionare un elenco di numeri casuali generati dinamicamente all'interno di una casella combinata (campo <SELECT>). Man mano che l'utente ne seleziona uno, lo script richiama se stesso e mostra il totale di elementi selezionati includendoli in un secondo elenco.

14 Crea una pagina Web che consenta di eseguire il gioco degli 11 fiammiferi utilizzando i cookies per memorizzare i dati parziali.

15 Crea un programma in grado di leggere i dati anagrafici di un utente e iniziare il gioco delle 3 carte, chiedendo sotto a quale delle 3 carte si nasconde l'asso.

I file e l'upload in php

IN QUESTA LEZIONE IMPAREREMO...

- a gestire i file di testo in php
- a utilizzare le funzioni php per l'upload di file

MAPPA CONCETTUALE

fread()
fwrite()
lettura
scrittura
fclose()
file binari
array
associativo
chiusura
$_FILES[]
apertura
upload
fopen()
File in php
funzione
move_uploaded_
file()
modi
di apertura
Sola lettura
Lettura
e scrittura
Scrittura
Lettura
e scrittura
distruttiva
Aggiunta
Binari
"r"
"r+"
"w"
"w+"
"a"
"b"

I file in php

I file vengono gestiti in php con istruzioni analoghe a quelle del linguaggio. L'apertura di un file avviene ricorrendo alla funzione fopen(), con la quale al file aperto viene associato un puntatore che rappresenta l'identificatore del file per operazioni di lettura e scrittura.

```
$idmiofile=fopen("nome del file","modalità di apertura");
```

La chiusura avviene con la funzione fclose(), indicando come parametro il puntatore relativo al file.

Il nome del file è rappresentato dal percorso del file da aprire, mentre la modalità di apertura può essere una delle seguenti:
- r sola lettura
- r+ lettura e scrittura
- w sola scrittura: il contenuto viene perso (*)
- w+ lettura e scrittura, ma perdendo il contenuto (*)
- a solo per aggiunta (modalità append) (*)
- a+ per lettura e aggiunta (*)
- b da aggiungere ai precedenti per trattare il file come file binario

(*) Se il file non esiste php tenta di crearlo.

Per verificare se l'apertura del file è andata a buon fine, è necessario controllare il contenuto del puntatore al file: se esso è falso significa che l'apertura è fallita.

```
$idfile=fopen("accessi.txt","r");
// Se l'apertura del file fallisce la funzione die() visualizza il messaggio e chiude lo script
if (!$idfile)
 die ("Si è verificato un errore durante l'apertura del file");
```

Di seguito, riportiamo le principali funzioni per la gestione dei files in php.

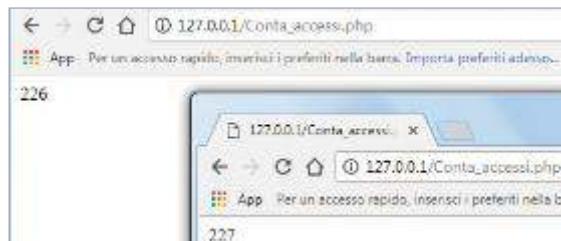
```
fopen($filedaaprire, $mode)                      // Apre un file
fread($filedaleggere)                            // Legge da un file
unlink($filedaeliminare)                         // Elimina un file
file_exists($filedacontrollare)                  // Controlla se un file esiste
is_writable($filedacontrollare)                  // Controlla se un file è riscrivibile
is_readable($filedacontrollare)                  // Controlla se un file è leggibile
fwrite($testodascrivere, $fileincuiscrivere)     // Scrive in un file eliminandone il contenuto precedente
fsize($filedacontrollare)                        // Restituisce la dimensione del file in byte
fclose($filedachiudere)                          // Chiude un file
```

La funzione che consente di leggere un file è fread(), che possiede due parametri, identificatore del file e numero di byte da leggere. Riportiamo qui di seguito un codice di esempio per leggere da un file contaccessi.txt i primi "*n*" caratteri:

```
// Apertura file in lettura
$idmiofile = fopen("/percorso/contaccessi.txt","r");
$n=100;
// Lettura di cento caratteri
$datiletti = fread($idmiofile,$n);
```

La funzione che permette di scrivere all'interno di un file è fwrite(). Possiede due parametri, il puntatore al file e la stringa da scrivere. Per scrivere una riga e andare a capo è necessario far terminare la stringa con i caratteri di escape \r\n, come in questo esempio:

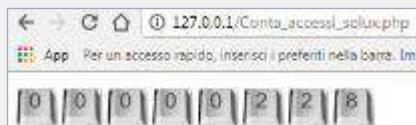
```
// Apertura file in lettura e scrittura con percorso relativo
// In questo caso il file deve trovarsi nella cartella del codice
$id = fopen("contaccessi.txt","w");
// Scrive la stringa indicata nel file mandando il testo a capo
$bytescritti = fwrite($id,"testo \r\n");
```

Vediamo di seguito un esempio in cui viene utilizzato un file per memorizzare il numero di accessi a una determinata pagina Web.

ESEMPIO

CONTATORE ACCESSI CON FILE

La prima operazione che viene effettuata per memorizzare il numero di accessi tramite file riguarda la verifica dell'esistenza del file mediante la funzione file_exists() (riga 5). Se il file non esiste, viene creato memorizzando 0 al suo interno (righe 20-26). Quando il file esiste, lo script provvede ad aprire il file (riga 8), quindi, dopo aver verificato se si riesce ad aprirlo (riga 10), legge il valore in esso contenuto (riga 14). A questo punto, la variabile che contiene il totale di visite ($conta_accessi) viene incrementata (riga 29), quindi, dopo aver riaperto il file in modalità scrittura distruttiva (riga 31), viene scritto il nuovo valore (riga 35). Infine, viene visualizzato il totale sul browser (riga 39).

```
1  <?php
2  //nome de file utilizzato nello script per memorizzare il totale accessi
3  $nomefile ="accessi.txt";
4  //Verifica dell'esistenza del file con la funzione file_exists()
5  if (file_exists($nomefile))
6  {
7     //se esiste gia viene aperto in lettura
8     $idfile=fopen($nomefile,"r+");
9     //se non si riesce ad prirlo viene creato un messaggio di errore
10    if (!$idfile) die ($msg="il file &nomefile non è stato aperto<BR>");
11       //se il file viene aperto correttamente
12       //vengono letti i primi dieci caratteri e memorizzati nella variabile $conta_accessi
13       //il casting è d'obbligo per trasformare la stringa in intero
14       $conta_accessi = (int) fread($idfile,10);
15    //chiusura file
16    fclose($idfile);
17 }
18 else
19 {
20    //se il file non esiste viene aperto e creato contestualmente
21    $idfile=fopen($nomefile,"w+");
22    if (!$idfile) die ($msg="il file &nomefile non è stato aperto<BR>");
23       //se il file viene aperto correttamente viene inizializzata la variabile conta_accessi
24       $conta_accessi = 0;
25    //chiusura del file
26    fclose($idfile);
27 }
28 //incremento del contatore di accessi (conta_accessi)
29 $conta_accessi++;
30 //apertura del file in scrittura e lettura distruttiva
31 $idfile=fopen($nomefile,"w+");
32 if (!$idfile) die ($msg="il file &nomefile non è stato aperto<BR>");
33    //se file aperto correttamente
34    //scrittura nel file del contatore di accessi
35    fwrite($idfile,$conta_accessi);
36 //chiusura file
37 fclose($idfile);
38 //visualizzazione contatore accessi
39 echo($conta_accessi);
40 ?>
```

Per verificare il funzionamento dello script proviamo ad aprire più volte il browser sulla pagina php:

Puoi trovare il codice di questo esempio nel file Conta_accessi.php.

METTITI ALLA PROVA

➜ • Utilizzo dei tag HTML in php • Utilizzo delle funzioni di lettura e scrittura sui file

Apri il file Conta_accessi.php

1 Modifica il codice dell'esempio in modo tale che mostri dei numeri grafici al posto del numero per il totale delle visite effettuate alla pagina. Per fare questo, utilizza le immagini (0.gif, 1.gif, ... 9.gif) presenti nella sottodirectory \cifre della cartella UDA 4 PHP del CD-ROM.

2 Confronta la tua soluzione con quella presente nel file Conta_accessi_solux.php.

L'array associativo $_FILES

La variabile predefinita $_FILES contiene i nomi e le informazioni che riguardano i file provenienti dal campo di tipo file di un Form. Ciò consente di effettuare l'upload di file dal client al server. In una comunicazione client/server che permetta di effettuare l'upload dei file, gli elementi fondamentali sono i seguenti:

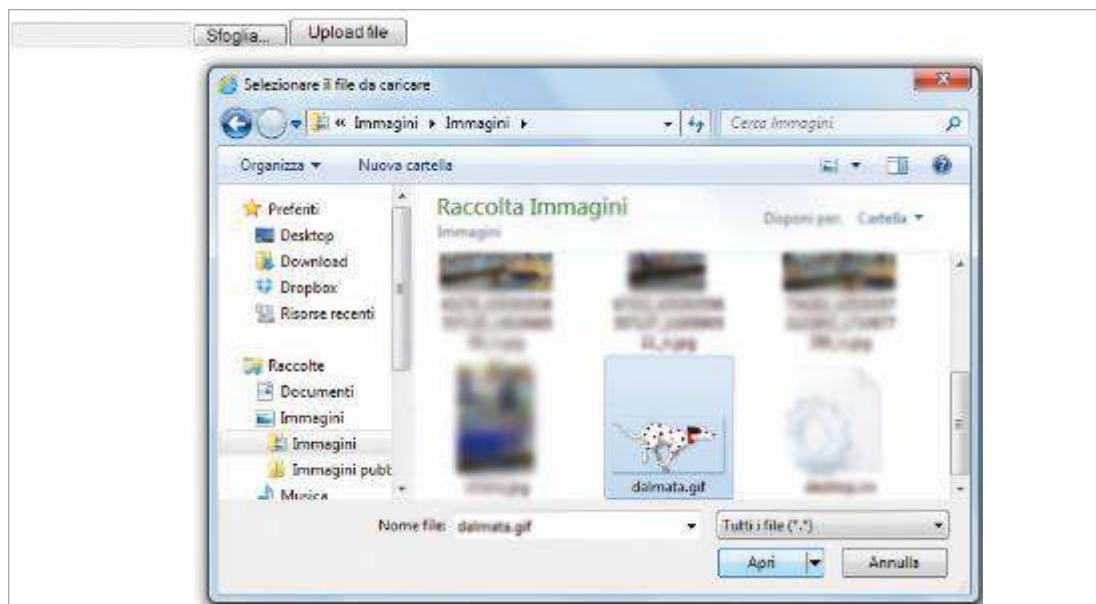
– Form HTML con campo file e proprietà enctype = multipart/form-data;
– funzione php move_uploaded_file(nome del file da caricare, percorso in cui caricarlo);
– array associativo $_FILES.

L'array $_FILES presenta la seguente struttura:

```
$_FILES[' nomecampofile '][”name”]       // nome e estensione del file caricato
$_FILES[' nomecampofile '][”type”]       // tipo di file caricato (in formato MIME type)
$_FILES[' nomecampofile '][”size”]       // dimensione del file caricato
$_FILES[' nomecampofile '][”tmp_name”]   // percorso completo del file temporaneo sul server
$_FILES[' nomecampofile '][”error”]      // codice numerico compreso fra 0 e 8 indicante il tipo di errore
```

Vediamo di seguito un esempio, che mostra come effettuare l'upload di un file di tipo immagine da un client a un server mediante un Form HTML.

ESEMPIO

UPLOAD DI FILE

Lo script di questo esempio si compone di due parti, secondo la tecnica postback (riga 3). La prima parte viene mostrata al primo accesso, quando cioè il campo $_POST['invia'] non è stato ancora settato (righe 4-10), e contiene un pulsante di invio (submit) e un campo di tipo file per selezionare il file da inviare al server.

La seconda parte racchiude lo script che viene elaborato alla pressione del tasto di submit, in questo caso Upload file. Innanzitutto i dati del file proveniente dal client vengono salvati nelle variabili di comodo (righe 14-16). Quindi viene verificato il tipo di file da caricare (riga 18). Se il tipo di file non è una immagine, viene segnalato un errore e lo script termina. Se il file da caricare è corretto viene usata la funzione move_uploaded_file, che carica effettivamente il file dal client al server nel percorso specificato come secondo parametro: in questo caso C:\www.

```
1  <?php
2  //Verifica campo - tecnica POSTBACK -
3  if (!isset($_POST['invia']))
4  {
5      //viene mostrato il form
6      echo "<FORM ACTION='".$_SERVER['PHP_SELF']."' ENCTYPE='multipart/form-data' METHOD='post'>";
7      echo "<INPUT TYPE='file' NAME='file_caricato'>";
8      echo "<INPUT TYPE='submit' value='Upload file' NAME='invia'>";
9      echo "</FORM>";
10 }
11 else
12 {
13     //trasferisco il tipo, il percorso completo (nome_tmp) e il nome del file in 3 variabili di comodo
14     $f=$_FILES['file_caricato']['type'];
15     $nome=$_FILES['file_caricato']['name'];
16     $nome_tmp=$_FILES['file_caricato']['tmp_name'];
17     //verifico il tipo di file, se si tratta di un'immagine (jpg, gif o png)
18     if (($f=="image/jpeg")||($f=="image/gif")||($f=="image/png"))
19     {
20         //Con move_uploaded_file il file verrà caricato dal client al server
21         //nel percorso specificato come secondo parametro
22         move_uploaded_file($nome_tmp,"C:/www/$nome");
23     }
24     else
25     {
26         //se tipo file non consentito
27         echo "Estensione non consentita";
28     }
29 }
30 ?>
```

Facendo click su Sfoglia... andiamo a selezionare il file da caricare sul server (in questo caso, il file dalmata.gif):

Dopo aver confermato con il pulsante Upload file che invia il Form al server, otteniamo (sul server), all'interno della cartella indicata come secondo parametro della funzione move_uploaded_file(), il file desiderato nella cartella C:\www:

Puoi trovare il codice di questo esempio nel file Upload.php.

METTITI ALLA PROVA

➜ • Applicazione delle sessioni e dei file • Utilizzo delle funzioni di upload

Apri il file Download.php

1 Il programma carica i file all'interno della sottodirectory ./programmi. Indenta il codice e modificalo in modo tale che si possano caricare i file in due cartelle (programmi e lavori).
2 Inserisci un secondo contatore che conti i download effettuati dai file della cartella lavori.

VERIFICA... le conoscenze

SCELTA MULTIPLA

1 Per creare un file, il codice corretto è:

a `$idfile=fopen ("miofile.txt","w+");`
b `$idfile=fopen ("miofile.txt","c");`
c `$idfile=fopen ("miofile.txt","c+");`
d `$idfile=fopen ("miofile.txt","w");`

2 La funzione fsize():

a ritorna la dimensione del file in Kbyte
b ritorna la dimensione del file in Mbyte
c ritorna la dimensione del file in bit
d ritorna la dimensione del file in byte

2 Il puntatore al file è:

a un indirizzo di memoria
b un numero intero
c un riferimento all'inizio del file
d nessuna delle risposte è corretta

3 La modalità di apertura di un file w+:

a è di lettura e scrittura
b è di lettura
c è di scrittura
d è di lettura e scrittura, ma perdendo il contenuto

4 Il comando $datiletti = fread($idfile,6);:

a legge il puntatore del file
b legge i primi 6 caratteri dell'ID_file
c legge i primi 6 caratteri del file
d legge 6 caratteri del file a partire dalla posizione corrente

5 Il comando $bytescritti = fwrite($idfile,"ali baba");:

a provoca un errore
b scrive "ali baba" all'inizio del file indicato
c in $bytescritti assegnerà valore 8
d scrive "ali baba" all'inizio del file indicato solo se aperto in mutua esclusione

6 Il comando fread($idfile);:

a restituisce TRUE se il file è terminato
b restituisce FALSE se il file è terminato
c restituisce 0 se il file è terminato
d restituisce –1 se il file è terminato

7 Il comando feof($idfile);:

a restituisce TRUE se il file è terminato
b restituisce FALSE se il file è terminato
c posiziona il file a EOF
d chiude il file

8 Il comando fseek($idfile,10);:

a legge 10 byte dal file
b si posiziona al 10 byte del file
c "sposta in avanti" la posizione corrente di 10 byte
d cerca il valore 10 nel file

9 La funzione rewind($idfile):

a riposiziona a 0 la posizione corrente
b riposiziona a 1 la posizione corrente
c è sintatticamente errata
d resetta in scrittura il file indicato

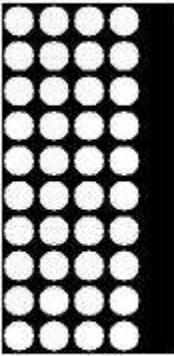
10 La variabile predefinita $_FILES contiene:

a i nomi dei file più recenti caricati sul server
b le informazioni sui file da caricare presenti nei campi TEXT di un Form
c le informazioni provenienti dal campo file di un Form
d i nomi dei file di cui effettuare l'upload

11 L'istruzione:

```
move_uploaded_file($_FILES['file'],"./
files/file.dat");
```

a carica sul server nel percorso indicato come primo parametro il file presente come secondo parametro
b legge dal Form il nome del file da caricare (secondo parametro)
c legge dal Form il nome del file da caricare (primo parametro)
d carica sul server nel percorso indicato come secondo parametro il file presente come primo parametro

VERIFICA... le competenze

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Mostra un Form composto da due list box, regione e provincia di nascita, quindi trasferiscili in una seconda pagina memorizzandoli in un file.

2 Scrivi uno script che verifichi la presenza di un file inserito dall'utente e, se presente, lo carichi sul server. Se non presente, lo deve creare scrivendoci al suo interno il tuo nome.

3 Scrivi uno script che legga il contenuto del file nani.txt e lo mostri a video, all'interno di una tabella con righe a colori di sfondo alternati.

4 Scrivi uno script che legga il contenuto del file nani.txt e lo riscriva a video alla rovescia.

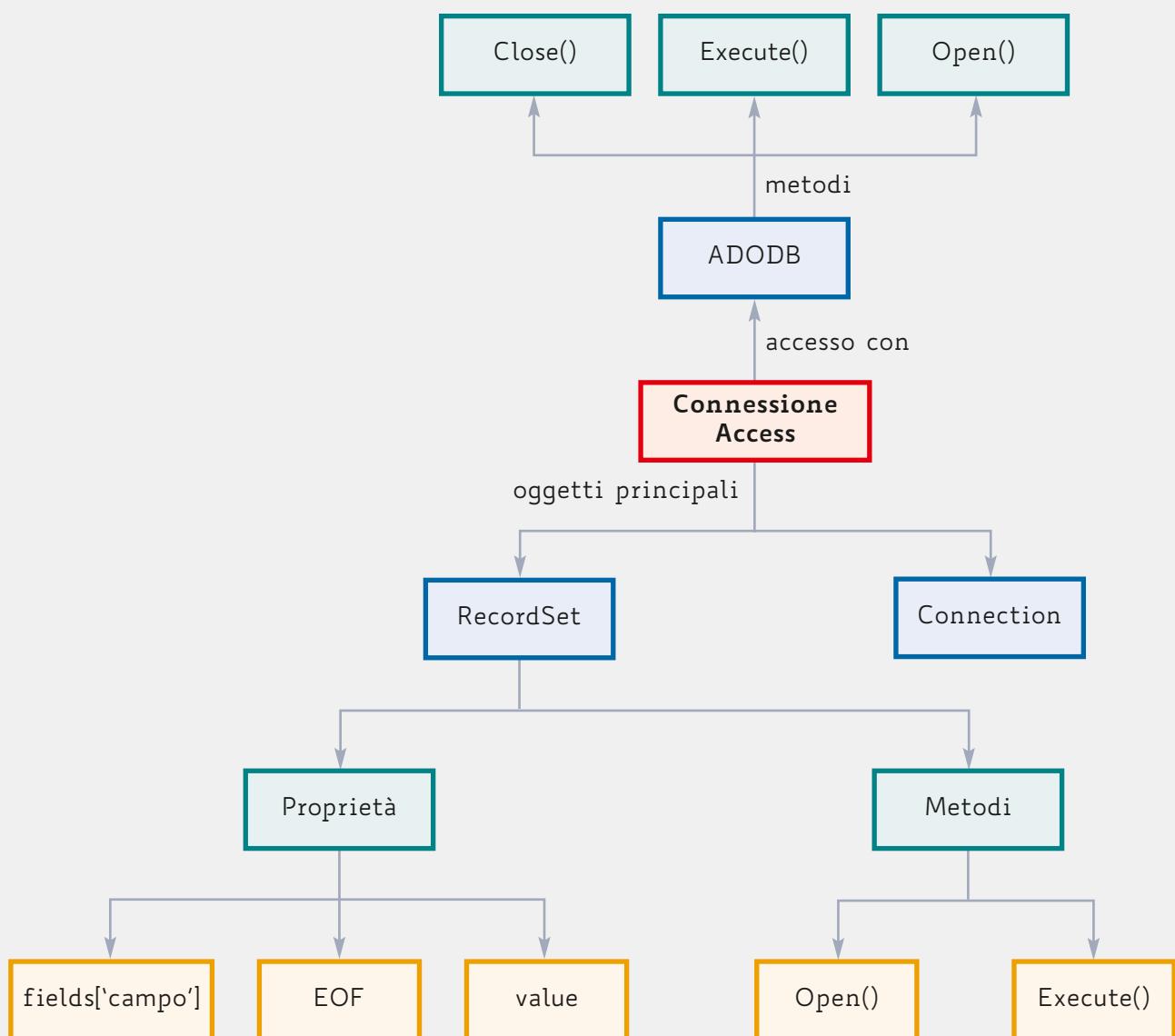
5 Scrivi uno script che mostri a video il nome, il percorso, l'estensione, la data e l'ora di creazione del file nani.txt.

6 Visualizza il contenuto di una cartella a piacere. Per ogni file (di testo), crea un link che consenta di visualizzarne il contenuto.

7 Crea *N* numeri causali e memorizzali su di un file di testo numeri.txt. Stampa quindi a video i numeri pari e dispari in due colonne all'interno di una tabella.

8 Crea uno script che mostri l'elenco dei file della cartella /programmi e consenta all'utente di caricarne uno sul server.

9 Crea uno script per simulare il gioco di forza4 con il computer, utilizzando le sessioni e i file per memorizzare i risultati parziali dei due giocatori.

10 Crea uno script per simulare il gioco del MasterMind utilizzando i file per memorizzare le soluzioni e le impostazioni, come indicato nella seguente figura:

La connessione al database Access

IN QUESTA LEZIONE IMPAREREMO...

- a gestire i dati di Access da php con connessione ADODB
- a eseguire alcuni comandi SQL da php
- a leggere, scrivere, aggiornare e cancellare dati da Access

MAPPA CONCETTUALE

didattica inclusiva

Close()
Execute()
Open()
metodi
ADODB
accesso con
Connessione Access
oggetti principali
RecordSet
Connection
Proprietà
Metodi
fields['campo']
EOF
value
Open()
Execute()

La connessione Access con ADODB

L'accesso ai dati presenti in un database Access può essere effettuato mediante ADODB (*ActiveX Data Objects Data Base*), una libreria che consente di gestire la connessione attraverso alcuni oggetti; i più importanti sono:
– Connection;
– RecordSet.

Li esaminiamo nei paragrafi che seguono.

L'oggetto Connection

L'oggetto preposto a fornire la connettività verso un database è di tipo Connection e va dichiarato nel modo seguente:

```
$connessione = new COM("ADODB.Connection");
```

In questo caso, tramite $connessione possiamo accedere a metodi e proprietà, come per esempio ConnectionString, che contiene le informazioni utilizzate per stabilire la connessione stessa.
I metodi principali sono:
– Open(), che apre una connessione;
– Execute(), che esegue una stringa contenente una istruzione SQL;
– Close(), che chiude l'oggetto Connection che è stato aperto.

Per aprire il database usiamo il metodo Open(), la cui sintassi è:

```
$connessione->Open($stringa_di_connessione);
```

Il parametro che vediamo tra parentesi ($stringa_di_connessione) contiene una particolare istruzione che attiva la connessione al database, chiamata stringa di connessione.
La stringa di connessione è costituita da una serie di coppie chiave=valore, separate tra di loro con un punto e virgola (;).
La situazione standard prevede che in tale stringa vengano inserite almeno due chiavi, che indicano:
– il formato di database utilizzato ("DRIVER=...");
– la collocazione fisica all'interno del file system del server ("DBQ=...").

Una tipica sequenza di comandi per un database in formato Access (esempio.mdb nel CD-ROM) è la seguente:

```
$percorso_database = realpath("./esempio.mdb");
$connessione = new COM("ADODB.Connection");
$stringa_di_connessione = "DRIVER={Microsoft Access
Driver(*.mdb)};DBQ=".$percorso_database;
$connessione->Open($stringa_di_connessione);
$recordset=$connessione->Execute($stringaSQL);
$connessione->Close();
```

È necessario convertire il percorso del file, collocato in questo caso nella stessa directory dello script, nel formato del sistema operativo, mediante la funzione realpath.

Mediante il metodo Execute viene eseguita una istruzione SQL memorizzata nella stringa passata come parametro. La sua sintassi è:

```
$recordset=$connessione->Execute($stringaSQL);
```

Infine, per chiudere la connessione utilizziamo il metodo Close:

AREA DIGITALE

Classe COM non presente

```
$connessione->Close();
```

L'oggetto RecordSet

Per creare un oggetto di tipo RecordSet dobbiamo usare l'operatore di istanza new e il costruttore COM, con la sintassi seguente:

```
$rs = new COM("ADODB.Recordset");
```

Per fare in modo che venga eseguita una interrogazione sul recordset utilizziamo il metodo Open:

```
$rs->Open($interrogazione,$connessione);
```

Il metodo Execute della connessione restituisce un riferimento a un oggetto di tipo RecordSet che rappresenta un insieme di tuple risultato del database. Per esempio, nell'istruzione seguente, l'oggetto $risul conterrà le tuple risultato presenti nella tabella Elenco del database:

```
$risul = $connessione.Execute("SELECT * FROM Elenco");
```

I recordset possono essere immaginati come vere e proprie tabelle di dati, in cui ciascuna riga rappresenta un record. I valori dei singoli campi del record evidenziato dal cursore sono leggibili tramite l'array associativo fields['nome campo'] dell'oggetto RecordSet che possiede la proprietà value che ne identifica il contenuto.
Il codice seguente mostra come leggere i record di un recordset per stamparli a video. Possiamo notare che il ciclo verifica innanzitutto che non siano finiti i record del recordset attraverso la proprietà EOF:

```
// Ciclo che termina quando EOF restituisce true
while (!$rs->EOF)
{
 // Visualizzazione del singolo campo Cognome
 echo $rs->fields['Cognome']->value.'<BR>';
 // Visualizzazione del singolo campo Nome
 echo $rs->fields['Nome']->value. '<BR>';
 // Visualizzazione del singolo campo Indirizzo
 echo $rs->fields['Indirizzo']->value. '<BR>';
 // Metodo Movenext() sposta il controllo al record successivo
 $rs->Movenext();
}
```

I principali metodi e proprietà dell'oggetto recordset sono elencati nella tabella sotto riportata.

Metodi	Proprietà
Delete	BOF
Find	EOF
Move	Index
MoveFirst	fields
MoveLast	value
MoveNext	
MovePrevious	
Open	
Seek	
Update	

Per effettuare le operazioni di ricerca, inserimento, modifica e cancellazione dei record esistono due diverse tecniche:
- utilizzo di una stringa che contiene i comandi SQL collocata come argomento del metodo Execute dell'oggetto Connection. In questo caso, si utilizza il recordset ottenuto solo per la visualizzazione dei risultati;
- creazione di un oggetto di tipo RecordSet, per poi effettuare su di esso le operazioni mediante i suoi metodi.

Nel primo caso è necessario creare oggetti di entrambi gli elementi (Connection e RecordSet), mentre nel secondo caso non è necessario utilizzare il recordset, in quanto viene creato automaticamente dall'esecuzione della query con il metodo Execute.

Lettura dati da Access

Per leggere una tabella di un database, dopo aver effettuato la connessione tramite ADODB, è necessario applicare una query SQL del tipo SELECT... FROM..., memorizzandola in una stringa da passare come parametro al metodo Execute del recordset.
Le tuple risultato vengono memorizzate nel recordset restituito dal metodo Execute stesso.

Vediamo un esempio.

ESEMPIO

LETTURA DATI DA ACCESS CON ADODB
In questo esempio vogliamo leggere l'intero contenuto di una tabella (clienti) del database clienti.mdb (presente nella sottocartella \mdb del CD-ROM). Per fare questo, dopo aver effettuato la connessione tramite ADODB (righe 3-5), eseguiamo la query di selezione (SELECT) mediante metodo Execute (riga 7). Le righe 9-14 consentono di generare il codice HTML necessario alla visualizzazione di una tabella, usata successivamente (righe 18-25) per mostrare in maniera ordinata i dati presenti nel recordset. Il ciclo while (riga 16) consente di effettuare la scansione del recordset $rs per stampare le tuple risultato presenti in esso.

```
1  <?php
2  //Connessione ADODB
3  $cn = new COM("ADODB.Connection");
4  $conn = "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=".realpath("./db/clienti.mdb").";";
5  $cn->Open($conn);
6  //Stringa di query per ricerca di tutti i record
7  $rs = $cn->Execute("SELECT * FROM clienti");
8  //Generazione codice HTML
9  echo ("<TABLE BORDER=1");
10 echo("Tabella Clienti");
11 echo("<THEAD><TR>");
12 //Visualizzazione elenco campi nella prima riga della tabella
13 echo("<TH>ID cliente<TH>Nome cliente<TH>Indirizzo<TH>CAP<TH>Citta'<TH>Provincia<TH>Partita
   IVA<TH>Codice fiscale");
14 echo("</TR></THEAD><TBODY>");
15 //Ciclo di lettura di tutti i record del recordset
16 while (!$rs->EOF)
17 {
18     echo "<TR><TD>".$rs->Fields('ID_cliente')->value."</TD>";
19     echo "<TD>".$rs->Fields("nome")->value."</TD>";
20     echo "<TD>".$rs->Fields("indirizzo")->value."</TD>";
21     echo "<TD>".$rs->Fields("cap")->value."</TD>";
22     echo "<TD>".$rs->Fields("citta")->value."</TD>";
23     echo "<TD>".$rs->Fields("provincia")->value."</TD>";
24     echo "<TD>".$rs->Fields("partita_iva")->value."</TD>";
25     echo "<TD>".$rs->Fields("codice_fiscale")->value."</TD></TR>";
26     //Record successivo
27     $rs->MoveNext();
28 }
29 echo("</TBODY></TABLE>");
30 //Chiusura connessioni
31 $rs->Close();
32 $cn->Close();
33 ?>
```

Il risultato dello script è il seguente:

127.0.0.1/Elenca.php

App Per un accesso rapido, inserisci i preferiti nella barra. Importa preferiti adesso...

ID cliente	Nome cliente	Indirizzo	CAP	Citta'	Provincia	Partita IVA	Codice fiscale
29	Malpex Srl	Via Cavour 5	22100	Milano	MI	896778989	
30	Tecnoplast SpA	Via Papa XXIII, 3	16100	Genova	GE	343212345	
2	Rossi & Bianchi Srl	Via Verdi 42	22100	Como	CO	123456789	
3	Martelli SaS	Via	88100	Catanzaro	CZ	345678912	mrtytg54f12f9

Puoi trovare il codice di questo esempio nel file Elenca.php.

Vediamo adesso un secondo esempio.

ESEMPIO

LETTURA DATI DA ACCESS CON ADODB E VISUALIZZAZIONE CSS

In questo esempio vogliamo leggere i dati contenuti in una tabella (Prodotti) del database supermercati.mdb (presente nella sottocartella \mdb del CD-ROM) mostrandoli a video tramite un foglio di stile CSS che ne migliora l'aspetto grafico (tabelle.css nel CD-ROM).

La variabile $percorso contiene il percorso relativo (rispetto alla posizione dello script) del database di Access che utilizzeremo per leggerne i record (riga 8). Viene quindi creata la stringa di connessione ADODB, che è salvata nella variabile $sc (riga 10), poi vengono dichiarati due oggetti di tipo Connection e RecordSet, chiamati rispettivamente $cn e $rs (righe 12-13). Attraverso il metodo Open() vengono adesso create, rispettivamente, la connessione (riga 15) e il recordset, al quale viene passata la stringa di interrogazione (riga 16). Viene quindi stampata la prima riga della tabella secondo i tag HTML (righe 18-19):

```
6  <?php
7  //percorso del database, in questo caso nella sottocartella db
8  $percorso = realpath("./db/supermercati.mdb");
9  //Stringa di connessione ADO DB
10 $sc="DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=".$percorso.";";
11 //Creo due oggetti COM contenenti gli oggetti Connection e Recordset
12 $cn = new COM("ADODB.Connection") or die("Non va ADO");
13 $rs = new COM("ADODB.Recordset");
14 //Apro la Connection ed il Recordset
15 $cn->Open($sc);
16 $rs->Open("SELECT * FROM Prodotti", $cn);
17 //Stampa tabella con riga di intestazione
18 echo "<TABLE><TR><TH>ID_prodotto<TH>Nome prodotto<TH>Scorta
19        <TH>Marca<TH>Fornitore<TH>Giacenza<TH>Imponibile<TH>Sconto %<TH>IVA%";
20 $alt = false;
21 //Ciclo di lettura e stampa del recordset
```

Esaminiamo ora la sezione di codice che effettua il ciclo di lettura del recordset. Il ciclo while (riga 21) possiede come condizione di uscita la fine del recordset, rappresentata dalla condizione sulla proprietà EOF del recordset stesso. Tale ciclo potrebbe essere interpretato nel modo seguente: "*mentre il recordset non è finito esegui...*".

La variabile $alt ha un ruolo importante nella formattazione, sulla base del foglio di stile, delle righe della tabella che saranno visualizzate. Questa variabile è infatti impostata a false e diventa true alternativamente. In tal modo, la variabile di comodo $altclass contiene la chiamata alla classe alt (class='alt') solo per le righe pari (riga 25). La tabella viene così stampata a due colori, per aumentarne la leggibilità.

Viene quindi mostrato il contenuto del singolo campo del record interessato (righe 28-36) mediante la proprietà value dell'array associativo fields, la cui chiave rappresenta il nome del campo del record presente nel recordset.

Il metodo MoveNext sposta il controllo al record successivo del recordset.

Infine, la variabile $alt viene negata per invertirne (negarne) il contenuto (riga 41).

È importante ricordarsi di chiudere la connessione per maggiore sicurezza (riga 45).

```
21 while (!$rs->eof)
22 {
23     //in base al contenuto della variabile $alt viene inserito lo stile altClass
24     //per ottenere l'effetto del colore di sfondo alternato nelle righe
25     $altClass = $alt ? " class='alt'" : "";
26     echo "<TR>";
27     //Stampa del singolo campo in ogni cella
28     echo'<TD'.$altClass.'>'.$rs->fields['id_prodotto']->value."</TD>";
29     echo'<TD'.$altClass.'>'.$rs->fields['nome']->value."</TD>";
30     echo'<TD'.$altClass.'>'.$rs->fields['scorta']->value;
31     echo'<TD'.$altClass.'>'.$rs->fields['marca']->value;
32     echo'<TD'.$altClass.'>'.$rs->fields['fornitore']->value;
33     echo'<TD'.$altClass.'>'.$rs->fields['giacenza']->value;
34     echo'<TD'.$altClass.'>'.$rs->fields['imponibile']->value;
35     echo'<TD'.$altClass.'>'.$rs->fields['sconto%']->value;
36     echo'<TD'.$altClass.'>'.$rs->fields['iva%']->value;
37     echo'</TR>';
38     //Metodo MoveNext() per passare al record successivo
39     $rs->MoveNext();
40     //negazione del contenuto di $alt per riga successiva
41     $alt = !$alt;
42 }
43 echo "</TABLE>";
44 //Chiusura connessione
45 $cn->close();
46 ?>
```

L'esecuzione dello script è mostrata a pagina seguente. Non sono stati riprodotti tutti i record ma solo alcuni a titolo di esempio.

ID_prodotto	Nome prodotto	Scorta	Marca	Fornitore	Giacenza	Imponibile	Sconto %	IVA%
1	calze uomo	10	uniplast	verdi	21	10	0	0.2
2	cozze surgelate	10	unigel	bianchi	11	7	0	0.04
3	prezzemolo	20	pastogel	verdi	18	1,5	0	0.04
4	spinaci	10	findus	bianchi	9	2,2	0	0.04
5	platessa	30	oceano	brutti	32	17.9	0.1	0.04
6	bastoncini pesce	10	findus	brutti	15	12,5	0	0.04

Puoi trovare il codice di questo esempio nel file Adodb.php che utilizza il database supermercati.mdb.

Scrittura dati su Access

La scrittura di dati su di una tabella Access avviene, dopo aver effettuato la connessione ADODB al database, per mezzo del metodo Execute della connessione, al quale viene passato il comando INSERT di SQL. In questo caso, non è necessario istanziare un oggetto recordset in quanto non vi sono tuple risultato.

Vediamo un esempio.

ESEMPIO

SCRITTURA RECORD SUL DATABASE ACCESS CON ADODB

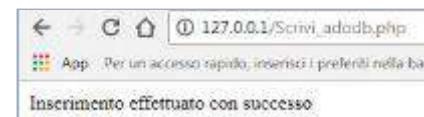
Utilizziamo la tecnica postback per leggere i dati di un Form, da inserire nella tabella Prodotti del database supermercati.mdb, usato anche nell'esempio precedente.

Analizziamo il codice dello script: innanzitutto la tecnica postback prevede che venga testato un campo qualsiasi ricevuto dal Form attraverso l'istruzione isset($_POST['nome']), per verificare se si tratta del primo accesso. In caso negativo (riga 3), viene mostrato il Form all'utente (righe 6-18):

```
1  <?php
2  //Tecnica POSTBACK per realizzare il form nella stessa pagina
3  if (!isset($_POST['nome']))
4  {
5     //Stampa del form
6     $form="<FORM ACTION='".$_SERVER['PHP_SELF']."' METHOD=POST><TABLE><TR>
7            <TD>Nome prodotto<TD><INPUT TYPE='text' NAME='nome'>
8     <TR><TD>Scorta<TD><INPUT TYPE='text' NAME='scorta'>
9     <TR><TD>Marca prodotto<TD><INPUT TYPE='text' NAME='marca'>
10    <TR><TD>Fornitore<TD><INPUT TYPE='text' NAME='fornitore'>
11    <TR><TD>codice del reparto<TD><INPUT TYPE='text' NAME='cod_rep'>
12    <TR><TD>Giacenza<TD><INPUT TYPE='text' NAME='giacenza'>
13    <TR><TD>Imponibile<TD><INPUT TYPE='text' NAME='imponibile'>
14    <TR><TD>Sconto percentuale<TD><INPUT TYPE='text' NAME='sconto'>
15    <TR><TD>IVA percentuale<TD><INPUT TYPE='text' NAME='iva'>
16    <TR><TD>Prezzo<TD><INPUT TYPE='text' NAME='prezzo'>
17    <TR><TD><INPUT TYPE=submit VALUE='scrivi record'></TABLE>";
18    echo $form;
19 }
20 else
```

La seconda parte dello script riceve i dati dal Form. La variabile $percorso contiene il percorso relativo (rispetto alla posizione dello script) del database di Access che utilizzeremo per leggerne i record (riga 23). Viene quindi creata la stringa di connessione ADODB, che è salvata nella variabile $sc (riga 24), quindi viene dichiarato l'oggetto Connection, ma non il RecordSet (riga 25): in questo caso vogliamo infatti utilizzare il metodo Execute per fare eseguire la query di inserimento record (INSERT) senza istanziare un recordset, dato che deve essere eseguita una query SQL che non restituisce nessuna tupla risultato.

Attraverso il metodo Open() viene creata la connessione (riga 26). La query di inserimento viene salvata nella variabile $sql (righe 28-33).

È necessario porre molta attenzione ai valori dei dati che intendiamo inserire: i campi di tipo numerico infatti non devono essere racchiusi tra apici, mentre i campi testuali necessitano di essere racchiusi tra apici.

Al metodo Execute passiamo come argomento la stringa di query per la sua esecuzione (riga 34).

```
21  - {
22        //Connessione ADODB
23        $percorso = realpath("./db/supermercati.mdb");
24        $sc="DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=".$percorso.";";
25        $cn = new COM("ADODB.Connection") or die("Non va ADO");
26        $cn->Open($sc);
27        //Definizione query di inserimento
28        $sql = "INSERT INTO
      Prodotti(nome,scorta,marca,fornitore,cod_reparto,giacenza,imponibile,[sconto%],[iva%],prezzo)
      VALUES (";
29        $sql.="'".$_POST['nome']."',".$_POST['scorta'].",'";
30        $sql.=$_POST['marca']."','".$_POST['fornitore']."',";
31        $sql.=$_POST['cod_rep'].",".$_POST['giacenza'].",";
32        $sql.=$_POST['imponibile'].",".$_POST['sconto'].",";
33        $sql.=$_POST['iva'].",".$_POST['prezzo'].");";
34        $cn->Execute($sql);
35        echo "<P>Inserimento effettuato con successo</P>";
36        // Chiudo la Connection
37        $cn->Close();
38        $cn = null;
39    }
40    ?>
```

L'esecuzione dello script è riportata di seguito: possiamo notare che, per i campi percentuali, i valori sono riportati come decimali:

Dopo aver fatto clic sul pulsante che invia il Form, viene mostrato il messaggio di avvenuto inserimento:

Possiamo quindi verificare l'avvenuto inserimento del record nella tabella del database aprendo il file di Access supermercati.mdb:

prodotti

ID_prodotto	nome	scorta	marca	fornitor	cod_	giacen	imponibile	sconto%	iva%	prezzo
43	Custodia Tablet	20	Dikom	mattei	10	15	€ 22,50	0,00%	22,00%	€ 0,00

Puoi trovare il codice di questo esempio nel file Scrivi_adodb.php che utilizza il database supermercati.mdb.

Modifica dati di Access

La modifica dei dati di una tabella Access avviene, dopo aver effettuato la connessione ADODB al database, per mezzo del metodo Execute della connessione, al quale viene passata, come parametro, la stringa SQL contenente il comando UPDATE.
Vediamo un esempio.

ESEMPIO

GESTIONE DI UNA TABELLA DI ACCESS DA PHP CON ADODB

In questo esempio utilizziamo il database clienti.mdb, presente nella sottocartella \mdb del CD-ROM. Il codice php prevede, nella prima sezione, la connessione tramite ADODB (righe 3-5) e la visualizzazione completa dei dati contenuti nei campi della tabella clienti (righe 7-25). Nella riga 16 viene mostrata una icona di modifica (ico.png) come collegamento ipertestuale in postback alla pagina stessa. Il collegamento ipertestuale è rappresentato da una query string basata sul campo GET di nome id al quale viene associato il campo ID_cliente del recordset ($rs->Fields('ID_cliente')->value).

```
1   <?php
2   //Connessione ADODB
3   $cn = new COM("ADODB.Connection");
4   $conn = "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=".realpath("./db/clienti.mdb").";";
5   $cn->Open($conn);
6   //Stringa di query per ricerca di tutti i record
7   $rs = $cn->Execute("SELECT * FROM clienti");
8   //Generazione codice HTML
9   echo("Tabella Clienti");
10  echo ("<TABLE BORDER=1><TH>");
11  echo("ID cliente<TH>Nome cliente<TH>Indirizzo<TH>CAP<TH>Citta'<TH>Provincia<TH>Partita IVA<TH>Codice fiscale<TR>");
12  //Ciclo di lettura di tutti i record del recordset
13  while (!$rs->EOF)
14  {
15      //Inserimento icona che chiama in postback la pagina per modifica record
16      echo "<TD><A HREF='?id=".$rs->Fields('ID_cliente')->value."'><IMG SRC='ico.png'></A></TD>";
17      echo "<TD>".$rs->Fields("nome")->value."</TD>";
18      echo "<TD>".$rs->Fields("indirizzo")->value."</TD>";
19      echo "<TD>".$rs->Fields("cap")->value."</TD>";
20      echo "<TD>".$rs->Fields("citta")->value."</TD>";
21      echo "<TD>".$rs->Fields("provincia")->value."</TD>";
22      echo "<TD>".$rs->Fields("partita_iva")->value."</TD>";
23      echo "<TD>".$rs->Fields("codice_fiscale")->value."</TD></TR>";
24      $rs->MoveNext();
25  }
26  echo("</TBODY></TABLE>");
```

La sezione successiva del codice mostra cosa accade quando la pagina riceve i dati in postback facendo clic sul link creato nella riga 16 della sezione precedente. Per prima cosa viene verificata l'esistenza (riga 28) del campo GET di nome id. Se il campo id è stato settato, significa che l'utente ha cliccato sull'icona di modifica, quindi viene eseguita la query di selezione con proiezione per estrarre i dati del database relativi al record con campo ID_cliente uguale a quello selezionato in precedenza (righe 31-33). Viene poi mostrato l'elenco dei campi sotto forma di Form di immissione (righe 35-44) con destinazione (ACTION) alla pagina stessa (riga 35):

```
27  //Verifica campo id (postback)
28  if (isset($_GET['id']))
29  {
30      //Visualizzazione form modifica record
31      $id = intval($_GET['id']);
32      //Query ricerca cliente in base alla chiave primaria
33      $rs = $cn->execute("SELECT * FROM clienti WHERE ID_cliente=".$id.";");
34      //Visualizzazione form
35      echo "<FORM ACTION='Modifica.php' METHOD='POST'>";
36      echo "ID Cliente <INPUT TYPE=text NAME=ID_cliente VALUE=".$rs->Fields('ID_cliente')->value."><BR>";
37      echo "Nome cliente <INPUT TYPE='text' NAME='nome' VALUE='".$rs->Fields("nome")->value."'><BR>";
38      echo "Indirizzo <INPUT TYPE='text' NAME='indirizzo' VALUE=".$rs->Fields('indirizzo')->value."><BR>";
39      echo "CAP <INPUT TYPE='text' NAME='cap' VALUE=".$rs->Fields('cap')->value."><BR>";
40      echo "Citta' <INPUT TYPE='text' NAME='citta' VALUE=".$rs->Fields('citta')->value."><BR>";
41      echo "Provincia <INPUT TYPE='text' NAME='provincia' VALUE=".$rs->Fields('provincia')->value."><BR>";
42      echo "Partita IVA <INPUT TYPE='text' NAME='partita_iva' VALUE=".$rs->Fields('partita_iva')->value."><BR>";
43      echo "Cod.fiscale <INPUT TYPE='text' NAME='codice_fiscale' VALUE=".$rs->Fields('codice_fiscale')->value."><BR>";
44      echo "<INPUT TYPE='submit' NAME='azione' VALUE='modifica'>";
45  }
```

La terza sezione di codice è relativa alla modifica dei dati nel database. Per prima cosa viene verificata l'esistenza del campo POST di nome azione (riga 46). Se il campo esiste e contiene il valore "modifica" significa che l'utente ha inserito dei dati nel Form e ha fatto clic sul pulsante di invio.
I campi provenienti in POST dal Form di immissione vengono letti in un ciclo foreach (righe 51-64) e

assegnati alla stringa che contiene la query SQL: vengono inoltre tolti i caratteri speciali eventualmente presenti e assegnati gli apici e le virgole attorno ai nomi dei campi. La query SQL viene completata (righe 66-67) ed eseguita (riga 68). Infine la funzione header (riga 71) richiama la pagina stessa.

```
45  }
46  if (isset($_POST['azione']) && $_POST['azione']=="modifica")
47  {
48      //Stringa di query iniziale
49      $SQL = "UPDATE clienti SET ";
50      //Ciclo per tutti i campi POST ricevuti dalla pagina
51      foreach($_POST as $key => $valori)
52      {
53          if ($key<>"ID_cliente" && $key<>'azione')
54          {
55              if (!empty($valori)) {
56                  //Generazione valori da campi POST
57                  //Funzione stripslashes ripulisce campo
58                  $valori = str_replace("'","''",stripslashes($valori));
59                  $SQL .= $key."='".$valori."', "; }
60              else {
61                  //Assegnazione NULL a campo ev. vuoto
62                  $SQL .= $key."=NULL, "; }
63          }
64      }
65      //Eliminazione ultima virgola
66      $SQL = substr($SQL, 0, -2);
67      $SQL .= " WHERE ID_cliente = ".$_POST['ID_cliente'];
68      $rs = $cn->Execute($SQL);
69      $cn->close();
70      //Richiamo pagina precedente che stampa elenco aggiornato
71      header("Location: ./Modifica.php");
72  }
```

L'esecuzione dello script mostra l'elenco dei record presenti nella tabella. Facendo clic sull'icona del record che si vuole modificare, appare una finestra di immissione per modificare i dati presenti. Infine il pulsante modifica completa l'operazione: i dati immessi vengono collocati nel database.

ID cliente	Nome cliente	Indirizzo	CAP	Citta'	Provincia	Partita IVA	Codice fiscale
	Malpex Srl	Via Cavour 5	22100	Milano	MI	896778989	
	Tecnoplast SpA	Via Papa XXIII, 3	16100	Genova	GE	343212345	
	Rossi & Bianchi Srl	Via Verdi 42	22100	Como	CO	123456789	
	Martelli SaS	Via	88100	Catanzaro	CZ	345678912	mrtytg54f12f9

Puoi trovare il codice di questo esempio nel file Modifica.php che utilizza il database clienti.mdb.

Cancellazione dati di Access

La cancellazione dei dati di una tabella Access avviene, dopo aver effettuato la connessione ADODB al database, per mezzo del metodo Execute della connessione al quale viene passato come parametro il comando DELETE di SQL.

ESEMPIO

CANCELLAZIONE RECORD DAL DATABASE ACCESS CON ADODB

In questo esempio viene utilizzato il database clienti.mdb, presente nella sottocartella \mdb del CD-ROM. Il codice php prevede, per prima cosa, la connessione tramite ADODB (righe 3-5) e la visualizzazione completa dei dati contenuti nei campi della tabella clienti (righe 7-25). Nella riga 16

viene mostrata una icona di cancellazione (delete.png) come collegamento ipertestuale in postback alla pagina stessa. Il collegamento ipertestuale è rappresentato da una query string basata sul campo GET di nome id, al quale viene associato il campo ID_cliente del recordset ($rs->Fields('ID_cliente')->value). Per verificare se la pagina è stata richiamata in postback viene controllata l'esistenza (riga 28) del campo GET di nome id. Se il campo id è stato settato, significa che l'utente ha cliccato sull'icona di cancellazione, quindi viene eseguita la query di cancellazione con campo ID_cliente uguale a quello selezionato in precedenza (righe 31-33). Infine la funzione header (riga 35) richiama la pagina stessa.

```
6   //Stringa di query per ricerca di tutti i record
7   $rs = $cn->Execute("SELECT * FROM clienti");
8   //Generazione codice HTML
9   echo("Tabella Clienti");
10  echo ("<TABLE BORDER=1><TH>");
11  echo("ID cliente<TH>Nome cliente<TH>Indirizzo<TH>CAP<TH>Citta'<TH>Provincia<TH>Partita IVA<TH>Cod
12  //Ciclo di lettura di tutti i record dal recordset
13  while (!$rs->EOF)
14  {
15      //Inserimento icona che chiama in postback la pagina per cancellazione record
16      echo "<TD><A HREF='?id=".$rs->Fields('ID_cliente')->value."'><IMG SRC='delete.png'></A></TD>";
17      echo "<TD>".$rs->Fields("nome")->value."</TD>";
18      echo "<TD>".$rs->Fields("indirizzo")->value."</TD>";
19      echo "<TD>".$rs->Fields("cap")->value."</TD>";
20      echo "<TD>".$rs->Fields("citta")->value."</TD>";
21      echo "<TD>".$rs->Fields("provincia")->value."</TD>";
22      echo "<TD>".$rs->Fields("partita_iva")->value."</TD>";
23      echo "<TD>".$rs->Fields("codice_fiscale")->value."</TD></TR>";
24      $rs->MoveNext();
25  }
26  echo("</TBODY></TABLE>");
27  //Verifica campo id (postback)
28  if (isset($_GET['id']))
29  {
30      //Stringa di query iniziale
31      $SQL = "DELETE FROM clienti WHERE ID_cliente = ".$_GET['id'];
32      $rs = $cn->execute($SQL);
33      $cn->close();
34      //Richiamo pagina precedente che stampa elenco aggiornato
35      header("Location: ./Cancella.php");
36  }
```

127.0.0.1/Cancella.php

Tabella Clienti

ID cliente	Nome cliente	Indirizzo	CAP	Citta'	Provincia	Partita IVA	Codice fiscale
	Malpex Srl	Via Cavour 5	22100	Milano	MI	896778989	
	Tecnoplast SpA	Via C.Colombo2	16100	Genova	GE	343212345	
	Rossi & Bianchi Srl	Via Verdi 42	22100	Como	CO	123456789	
	Martelli SaS	Via	88100	Catanzaro	CZ	345678912	mrtytg54f12f9

L'esecuzione dello script mostra l'elenco dei campi della tabella: facendo clic sull'icona di cancellazione verrà eliminato il record relativo.

Puoi trovare il codice di questo esempio nel file Cancella.php che utilizza il database clienti.mdb.

METTITI ALLA PROVA

➜ • Utilizzo di ADODB • Lettura, scrittura, modifica, cancellazione record di Access con php

Apri il file Scrivi_adodb.php

1 Unisci i due script (Adodb.php e Scrivi_adodb.php) in modo che l'utente possa selezionare se visualizzare o aggiungere record alla tabella.
2 Aggiungi uno script in grado di cancellare un record in base all'ID del prodotto.
3 Aggiungi uno script in grado di aggiornare il contenuto della tabella effettuando una ricerca del record da modificare in base all'ID_prodotto, quindi mostrando i dati del record trovato all'interno di un Form in modo che l'utente possa modificarli.

VERIFICA... le competenze

AREA DIGITALE

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Crea un database Access per la gestione della raccolta dei punteggi realizzati da un utente in alcuni videogiochi. Le pagine Web in php sono le seguenti:
- un Form in cui l'utente può inserire il nome del videogioco (una stringa) e il suo punteggio numerico. Il Form permette l'inserimento di un punteggio per volta e l'inserimento di più punteggi avviene inviando più volte il Form in una stessa sessione;
- una pagina di raccolta dei punteggi dei videogiochi che memorizza sul server i punteggi inseriti col Form del punto precedente. Si assume che se l'utente invia più di una volta un punteggio per lo stesso videogioco nella stessa sessione, il punteggio memorizzato è uguale al massimo dei valori inseriti;
- una pagina di riepilogo che stampa la lista dei videogiochi con i punteggi, il massimo punteggio e il videogioco che ha il punteggio massimo. Nel caso ci siano più videogiochi a cui corrisponde il punteggio massimo, si stampa il primo nella lista.

2 Crea un database Access per la gestione della raccolta dei risultati delle partite di calcio. Le pagine Web in php sono le seguenti:
- un Form in cui l'utente può inserire il risultato della partita selezionando la squadra di casa e quella ospite da un ComboBox generato dinamicamente a partire dalla lista delle squadre e specificando i gol delle due squadre in due campi di testo. Il Form permette l'inserimento di un risultato per volta e l'inserimento di più risultati avviene inviando più volte il Form in una stessa sessione;
- una pagina di raccolta dei risultati che memorizza sul server le partite inserite col Form del punto precedente. Si assume che se l'utente invia più di una volta un risultato per la stessa partita nella stessa sessione, il punteggio memorizzato è l'ultimo inserito;
- una pagina di riepilogo che stampa la classifica ordinata e calcolata assegnando 3 punti per la vittoria, 1 per il pareggio e 0 per la sconfitta.

3 Crea un database Access per la raccolta dei dati relativi alle temperature massime e minime di un insieme di stazioni meteorologiche. Le pagine Web in php sono le seguenti:
- un Form in cui l'utente può scegliere il nome della stazione da un menu a tendina e inserire la temperatura massima e minima. Il Form permette l'inserimento di una misura per volta e le opzioni del menu a tendina devono essere generate dinamicamente, a partire da un array con i nomi delle stazioni (si considerino "Trento", "Milano", "Torino", "Firenze", "Bologna", "Roma", "Napoli", "Bari", "Messina");
- una pagina di raccolta dei dati che memorizza sul server gli inserimenti fatti col Form di cui al punto precedente nella stessa sessione di lavoro. A ogni dato inserito deve essere associato anche il timestamp (si usi la funzione time(), che fornisce il timestamp come numero di secondi fra il tempo presente e la data del 1/1/1970 00:00:00);
- una pagina di riepilogo che stampa per ogni stazione la temperatura massima più alta e la media della temperatura minima del giorno corrente e dell'ultima settimana (si assuma che il giorno e la settimana corrente partano dal timestamp attuale meno 24*60*60 e 7*24*60*60 rispettivamente). Infine, supponendo che ciascuna stazione sia associata a una zona ("Trento" → "Nord", "Milano" → "Nord", "Torino" → "Nord", "Firenze" → "Centro", "Bologna" → "Centro", "Roma" → "Centro", "Napoli" → "Sud", "Bari" → "Sud", "Messina" → "Sud"), la pagina deve stampare la media, su tutti i dati disponibili, delle temperature massime e minime misurate per ciascuna zona.

La connessione al database MySQL

IN QUESTA LEZIONE IMPAREREMO...

- a gestire le tabelle e i tipi di dati di MySQL da php
- ad applicare la libreria mysqli

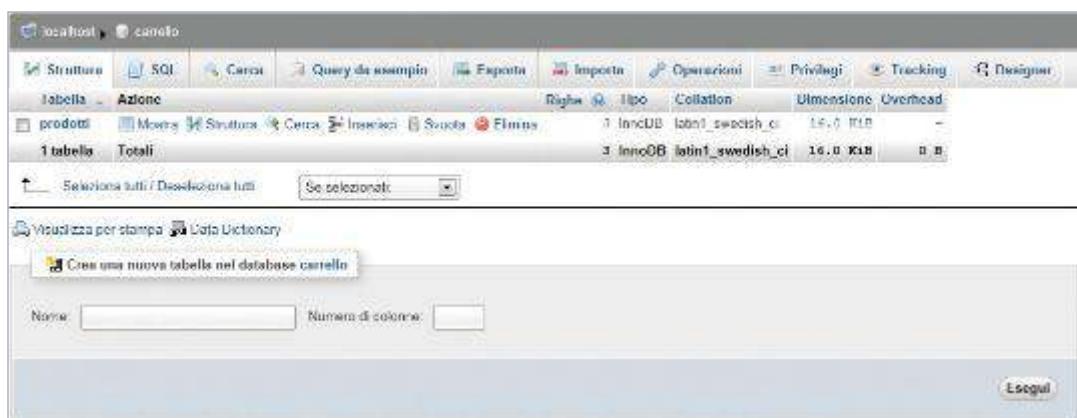
MAPPA CONCETTUALE

Connessione MySQL
accesso con
mysqli
oggetti principali
Resultset
Connection
metodo
query()
restituisce
Resultset
Proprietà
Metodi
num_rows
Array associativo
restituito da
Ciclo foreach
mysql_connect_errno()
mysql_connect_error()

Il DBMS MySQL

DBMS Server
Il compito di un DBMS server è quello di permettere a più richiedenti di accedere anche contemporaneamente alle medesime risorse, che potrebbero essere interi database o specifiche tabelle, o addirittura singoli record.

MySQL è un **DBMS Server** basato su SQL, multipiattaforma, relazionale, con licenza open source.

Per utilizzare MySQL dobbiamo innanzitutto attivare il servizio, facendo clic sul pulsante Running posto accanto a MySQL. Inoltre, per entrare nell'ambiente di amministrazione, chiamato phpMyAdmin, dobbiamo fare clic sul pulsante Admin, posto accanto al pulsante MySQL presente nel pannello di controllo di XAMPP:

Appare l'ambiente di amministrazione phpMyAdmin, dove possiamo accedere a un database selezionandolo nella colonna di sinistra, come mostrato nella figura a lato, dove viene selezionato un database di nome carrello.

La finestra che si apre consente di creare tabelle e database offrendo un ambiente di tipo GUI:

Le funzioni di connessione al database MySQL

Per effettuare la connessione da php con un database dobbiamo creare un oggetto mysqli() attraverso l'operatore new. La sintassi è la seguente:

```
$con = new mysqli("nomeHost","nomeUser","password","database")
```

In questo caso, $con è il nome dell'oggetto creato che rappresenta la connessione.

Il metodo connect_errno() consente di verificare successivamente se la connessione è avvenuta regolarmente: in particolare, restituisce true se la connessione è avvenuta senza problemi.

Riportiamo un breve codice illustrativo:

```
if (mysqli_connect_errno()) {
   echo("Errore di connessione");
   exit();
}
```

Possiamo ottenere il codice specifico dell'errore che si è eventualmente verificato mediante la funzione mysqli_connect_error().

> Al termine dello script bisogna rilasciare la risorsa, soprattutto per evitare problemi di sovraccarico del Web server, che altrimenti sarebbe costretto a lasciare un canale aperto a ogni connessione terminata. Per chiudere la connessione si utilizza il metodo $con->close().

Attraverso il metodo query() inviamo una query SQL, passata come parametro tra parentesi, al database aperto. La sua sintassi è la seguente:

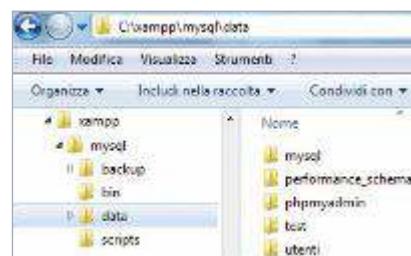
```
$ris = $con->query($stringa_query)
```

Il metodo query() restituisce un oggetto di tipo resultset ($ris), che rappresenta le tuple risultato. La proprietà num_rows contiene il numero di tuple risultato. Possiamo estrarre le tuple risultato dall'oggetto $ris mediante un ciclo foreach nel modo seguente:

```
// Verifica presenza di almeno una tupla risultato
if ($ris->num_rows > 0 ) {
   foreach($ris as $riga) {
      // L'array $riga contiene le tuple risultato estratte dall'oggetto $ris
      echo $riga['nome campo'];
      ...
      }
    }
```

Lettura dati da MySQL

Per leggere i dati da una tabella di un database MySQL, dopo aver effettuato la connessione tramite mysqli, è necessario applicare una query SQL del tipo SELECT... FROM... memorizzandola in una stringa da passare come parametro al metodo query. Le tuple risultato vengono memorizzate nell'oggetto resultset, restituito dal metodo query stesso.

Vediamo un esempio.

ESEMPIO

LETTURA DI RECORD DI UNA TABELLA MYSQL

Eseguiamo una selezione della tabella users, presente nel database utenti, con la clausola WHERE, per poi visualizzare i record che hanno conta_pres diverso da zero.

I database MySQL sono presenti nella directory di nome data, contenuta nella cartella mysql, presente sotto alla directory di xampp. Come possiamo notare, ciascuna directory contenuta in data rappresenta un database: in questo caso, possiamo individuare la cartella del database utenti.

Inizialmente avviene la connessione al database (righe 3-7). La query di selezione viene inserita in una variabile stringa ($sql) e poi inviata al server con il metodo query (righe 9-11). Possiamo verificare l'avvenuta esecuzione della query mediante l'istruzione or die, che comunica in caso di errore un messaggio e chiude lo script.
Il ciclo foreach (riga 15) estrae dal resultset ($ris) ogni tupla risultato della query nell'array associativo $riga, del quale vengono stampati i vari campi (righe 17-20). Infine, viene chiusa la connessione (riga 23).

```
1   <?php
2   //Creazione oggetto mysqli per realizzare la connessione
3   $con = new mysqli("localhost","root","","utenti");
4   if (mysqli_connect_errno()) {
5           echo("Connessione non effettuata: ".mysqli_connect_error()."<BR>");
6           exit();
7   }
8   //Definizione stringa contenente comando SQL
9   $sql = "SELECT ID_utente,nome_utente,password,conta_pres FROM users WHERE
    conta_pres<>0";
10  //Esecuzione query che restituisce $ris
11  $ris = $con->query($sql) or die ("Query fallita!");
12  //Genero tabella di visualizzazione
13  echo "<TABLE><TR><TH>ID utente<TH>Nome utente<TH>Password<TH>Contatore visite</TR>";
14  //Ciclo foreach legge gli elementi del resultset $ris
15  foreach($ris as $riga)
16  {
17      echo "<TR><TD>".$riga["ID_utente"];
18      echo "<TD>".$riga["nome_utente"];
19      echo "<TD>".$riga["password"];
20      echo "<TD>".$riga["conta_pres"];
21  }
22  //rilascio connessione
23  $con->close();
24  ?>
```

L'esecuzione dello script mostra l'elenco di record letti dalla tabella users:

ID utente	Nome utente	Password	Contatore visite
1	riccardo67	abcdef	41
2	paolo60	informatic	85
3	marco66	juve123	98
4	michela72	giugno04	12
5	bonnie63	bbb2634	71

Puoi trovare il codice di questo esempio nel file Elenca_mysql.php.

Scrittura dati su MySQL

Anche per aggiungere nuovi record in una tabella MySQL utilizziamo la stringa $sql, che contiene il comando SQL necessario per inserire record, rappresentato da INSERT. In questo caso, i valori posti tra parentesi nella clausola VALUES devono essere sostituiti dalle variabili del programma, come indicato nel codice di esempio seguente:

```
$sql = "INSERT INTO users (campo1, campo2, .... campo n) VALUES (".$val1.",
".$val2.",.....",".$valn.");";
```

La stringa viene poi passata come parametro al metodo query() per essere processata dal database.

È necessario porre molta attenzione alla creazione di questa query in quanto i valori dei campi testuali dovranno essere racchiusi tra singoli apici, mentre quelli dei campi numerici no.

Vediamo un esempio.

ESEMPIO

INSERIMENTO DI RECORD IN UNA TABELLA MYSQL

Realizziamo uno script che, utilizzando la tecnica postback, aggiunge un nuovo record nella tabella users. Alla riga 3 viene verificata l'esistenza del campo $_POST['invia']: se il campo è vuoto viene mostrato il Form (righe 4-9) necessario per far inserire all'utente i dati che verranno successivamente immessi nel database. Come possiamo notare, non vengono mostrate le caselle di immissione per i campi ID_utente e conta_pres: il primo in quanto campo di tipo auto_increment, il secondo in quanto assegnato inizialmente a zero.

```
1  <?php
2  //Se campo form non attivo (postback) mostra form
3  if (empty($_POST['invia'])) {
4      echo "<FORM ACTION='".$_SERVER['PHP_SELF']."' METHOD='POST'>";
5      echo "<TABLE><TR><TD>Nome utente<TD><INPUT TYPE='text' NAME='nome'>";
6      echo "<TR><TD>Password<TD><INPUT TYPE='text' NAME='pwd'>";
7      echo "<TR><TD><INPUT TYPE='submit' VALUE='Aggiungi Record' NAME='invia'>";
8      echo "</TABLE></FORM>";
9  }
```

La seconda parte dello script effettua prima di tutto la connessione al database (righe 11-15), poi azzera la variabile $conta e memorizza i campi POST 'nome' e 'pwd' nelle variabili $nome e $pwd (righe 17-19). La stringa contenente il codice SQL necessario per effettuare la INSERT viene creata nelle righe 21-22 ed eseguita nella riga 27. A questo punto viene mostrata la tabella aggiornata, in sostanza una ripetizione dell'esempio precedente in cui vengono mostrate tutte le righe della tabella (righe 28-34):

```
10 else {
11     //Connessione a MySQL
12     $con = new mysqli("localhost","root","","utenti");
13     if (mysqli_connect_errno()) {
14         echo("Connessione non effettuata: ".mysqli_connect_error()."<BR>");
15         exit();
16     }
17     $conta=0;
18     $nome=$_POST['nome'];
19     $pwd=$_POST['pwd'];
20     //Creazione query di comando sql
21     $sql = "INSERT INTO users(nome_utente,password,conta_pres)";
22     $sql.=" VALUES('".$nome."','".$pwd."',".$conta.");";
23     //Esecuzione query che restituisce $ris
24     $ris = $con->query($sql) or die ("Query fallita!");
25     //Visualizzazione tabella database aggiornata
26     $sql = "SELECT ID_utente,nome_utente,password,conta_pres FROM users;";
27     $ris = $con->query($sql) or die ("Query fallita!");
28     echo "<TABLE><TR><TH>ID utente<TH>Nome utente<TH>Password<TH>Contatore visite</TR>";
29     foreach($ris as $riga) {
30         echo "<TR><TD>".$riga["ID_utente"];
31         echo "<TD>".$riga["nome_utente"];
32         echo "<TD>".$riga["password"];
33         echo "<TD>".$riga["conta_pres"];
34     }
35     //Rilascio connessione
36     $con->close();
37 }
38 ?>
```

L'esecuzione dello script mostra un Form nel quale andiamo a immettere il nome utente e la password:

Dopo aver fatto clic su Aggiungi Record appare l'elenco aggiornato della tabella, in cui possiamo notare come il campo conta_pres contenga 0 e il campo ID_utente sia stato aggiornato automaticamente:

ID utente	Nome utente	Password	Contatore visite
1	riccardo67	abcdef	41
2	paolo60	informatic	85
3	marco66	juve123	98
4	michela72	michy	12
5	bonnie63	bbb2634	71
6	marione65	123456	0

Puoi trovare il codice di questo esempio nel file Inserisci_mysql.php.

Aggiornamento dati di MySQL

Anche in questo caso, utilizziamo la stringa di comando SQL per effettuare l'aggiornamento di un record mediante il comando SQL UPDATE, la cui sintassi è:

```
$sql = "UPDATE users SET campo1=".$val1.",campo2=".$val2.", ...
campon=".$valn." WHERE ID=".$id.";";
```

Vediamo un esempio.

ESEMPIO

AGGIORNAMENTO DI RECORD IN UNA TABELLA MYSQL

Realizziamo uno script in grado di ricercare un record, nella tabella users, attraverso il campo ID_utente, per poi mostrarlo in un Form in grado di consentirne la modifica. Tutto il codice è racchiuso nello stesso file php, utilizzando la tecnica postback. Il codice della pagina php può essere così suddiviso:

1 quando la pagina viene richiamata per la prima volta, mostra il Form di lettura campo ID per la ricerca;
2 quando la pagina viene richiamata in seguito all'inserimento del campo ID, effettua la ricerca del record nella tabella e mostra il record da modificare in un Form;
3 quando la pagina viene richiamata in seguito alla modifica del record avviene l'aggiornamento della tabella sul database.

Per verificare in quale delle tre situazioni sopra menzionate si trova il programma, viene effettuato un controllo sui campi POST di tipo submit chiamati 'invia' e 'modificato'. Se nessuno dei due è stato settato, siamo nella situazione del punto **1** (righe 3-8), se è settato solo 'invio' siamo nella situazione

del punto **2** (righe 14-36), se sono settati sia 'invio' che 'modificato' siamo nella situazione del punto **3** (righe 39-56).

Il codice relativo alla situazione descritta al punto **1** è il seguente:

```
1   <?php
2   //Se campo form non attivo (postback) mostra form
3   if (!isset($_POST['invia']) && !isset($_POST['modificato'])) {
4       echo "<FORM ACTION='".$_SERVER['PHP_SELF']."' METHOD='POST'>";
5       echo "<TABLE><TR>";
6       echo "<TD>ID_utente da modificare<TD><INPUT TYPE='text' NAME='id'>";
7       echo "<TR><TD><INPUT TYPE='submit' VALUE='Cerca Record' NAME='invia'>";
8       echo "</TABLE></FORM>";
9   }
```

Siamo invece nella situazione **2** se il campo POST 'modificato' non è stato settato, ma è stato settato 'invio': viene pertanto dapprima effettuata la connessione al database (righe 13-17), quindi, dopo aver salvato in $id il campo POST ricevuto e necessario per la ricerca (riga 19), viene memorizzata la stringa che contiene il comando SQL necessario per cercare il record (riga 20) e mandata in esecuzione (riga 21). Le righe successive mostrano a video il Form dei dati del record ottenuto (righe 22-36).

```
10  else
11  {
12      //Verifica se si tratta di ricerca o modifica
13      if (!isset($_POST['modificato'])) {
14          $con = new mysqli("localhost","root","","utenti");
15          if (mysqli_connect_errno()) {
16              echo("Connessione non effettuata: ".mysqli_connect_error()."<BR>");
17              exit();
18          }
19          $id=$_POST['id'];
20          $sql = "SELECT * FROM users WHERE ID_utente=".$id.";";
21          $ris = $con->query($sql) or die ("Query fallita!");
22          echo "<FORM ACTION='".$_SERVER['PHP_SELF']."' METHOD='post'>"
23          echo "<TABLE><TR><TH>ID utente<TH>Nome utente<TH>Password<TH>Contatore visite";
24          //Ciclo foreach legge gli elementi del resultset $ris
25          foreach($ris as $riga) {
26              echo "<TR><TD>ID<TD>";
27              echo "<INPUT TYPE='text' NAME='ID' VALUE=".$riga["ID_utente"].">";
28              echo "<TD>Nome<TD>";
29              echo "<INPUT TYPE='text' NAME='nome' VALUE=".$riga["nome_utente"].">";
30              echo "<TD>Password<TD>";
31              echo "<INPUT TYPE='text' NAME='password' VALUE=".$riga["password"].">";
32              echo "<TD>Accessi<TD>";
33              echo "<INPUT TYPE='text' NAME='pres' VALUE=".$riga["conta_pres"].">";
34          }
35          echo "<TR><TD><INPUT TYPE='submit' VALUE='MODIFICA' NAME='modificato'>";
36          echo "</TABLE></FORM>";
37      }
```

La parte successiva di codice rappresenta la terza fase, quella che provvede alla modifica vera e propria del record all'interno della tabella users. Prima di tutto viene effettuata nuovamente la connessione al database (righe 39-43), in quanto ogni volta che la pagina viene chiusa e riaperta vengono altresì chiuse tutte le connessioni attive. Vengono poi salvati i campi POST ricevuti, necessari in quanto verranno aggiornati all'interno della tabella (righe 45-48). Le righe 50-52 memorizzano la stringa SQL necessaria per aggiornare la tabella mediante il costrutto UPDATE ed eseguono la query stessa sul server.

È importante ricordare che, quando si inserisce un valore testuale all'interno della stringa della query SQL, è necessario racchiuderla tra apici singoli.

Infine viene mostrato un messaggio di successo che consente di richiamare la pagina stessa (riga 54).

```
38      else
39      {
40          $con = new mysqli("localhost","root","","utenti");
41          if (mysqli_connect_errno()) {
42              echo("Connessione non effettuata: ".mysqli_connect_error()."<BR>");
43              exit();
44          }
45          $id=$_POST['ID'];
46          $n=$_POST['nome'];
47          $pwd=$_POST['password'];
48          $conta=$_POST['pres'];
49          //Codice SQL per la modifica del record nella tabella
50          $sql = "UPDATE users SET nome_utente='".$n."',password='".$pwd."',conta_pres=";
51          $sql.= $conta." WHERE ID_utente=".$id.";";
52          $ris = $con->query($sql) or die ("Query fallita!");
53          $con->close();
54          echo "<A HREF='".$_SERVER['PHP_SELF']."'>Aggiornamento effettuato</A>";
55      }
56  }
57  ?>
```

L'esecuzione dello script mostra un Form da riempire con l'ID dell'utente da ricercare per la successiva modifica:

I dati dell'utente che è stato cercato vengono mostrati in un Form di modifica. Dopo averli modificati, facciamo clic su MODIFICA per confermare l'operazione:

Possiamo notare che, una volta effettuata la modifica, appare il messaggio di conferma:

Puoi trovare il codice di questo esempio nel file Aggiorna_mysql.php.

Cancellazione dati in MySQL

La cancellazione dei record avviene per mezzo del comando SQL DELETE. Potremmo sintetizzare in questo modo la sintassi utilizzata:

```
$sql = "DELETE FROM users WHERE ID_utente=".$campoid.";";
```

ESEMPIO

CANCELLAZIONE DI RECORD IN UNA TABELLA MYSQL

In questo esempio vogliamo realizzare uno script in grado di ricercare un record, nella tabella users, attraverso il campo ID_utente, per poi mostrarlo in un Form in grado di confermarne la cancellazione.

Tutto il codice è racchiuso nello stessp file php, utilizzando la tecnica postback. Il codice della pagina php può essere così suddiviso:

1 quando la pagina viene richiamata per la prima volta, mostra il Form di lettura campo ID per la ricerca;
2 quando la pagina viene richiamata in seguito all'inserimento del campo ID, effettua la ricerca del record nella tabella e mostra il record da cancellare;
3 quando la pagina viene richiamata in seguito alla conferma di cancellazione del record, avviene la cancellazione vera e propria del record nella tabella del database.

Per verificare in quale situazione si trova il programma, tra le 3 sopra menzionate, viene effettuato un controllo sui campi POST di tipo submit chiamati 'invia' e 'cancella'. Se nessuno dei due è stato settato siamo nella situazione del punto **1** (righe 3-8), se è settato solo 'invio' siamo nella situazione del punto **2** (righe 13-33), se sono settati sia 'invio' che 'cancella' siamo nella situazione del punto **3** (righe 35-47).

Il codice relativo alla situazione descritta al punto **1** è il seguente:

```
1  <?php
2  //Se campo form non attivo (postback) mostra form
3  if (!isset($_POST['invia']) && !isset($_POST['cancella'])) {
4      echo "<FORM ACTION='".$_SERVER['PHP_SELF']."' METHOD='post'>";
5      echo "<TABLE><TR>";
6      echo "<TD>ID_utente da Cancellare<TD><INPUT TYPE='text' NAME='id'>";
7      echo "<TR><TD><INPUT TYPE='submit' VALUE='cerca record' NAME='invia'>";
8      echo "</TABLE></FORM>";
9  }
```

Siamo invece nella situazione **2** se il campo POST 'cancella' non è stato settato ma è stato settato 'invio': viene pertanto dapprima effettuata la connessione al database (righe 13-17), quindi, dopo aver salvato in $id il campo POST ricevuto e necessario per la ricerca (riga 19), viene memorizzata la stringa che contiene il comando SQL necessario per cercare il record (riga 20) e mandata in esecuzione (riga 21). Le righe successive mostrano a video i campi del record ottenuto (righe 22-33).

```
10 else
11 {
12     //Verifica se si tratta di ricerca o cancellazione
13     if (!isset($_POST['cancella'])) {
14         $con = new mysqli("localhost","root","","utenti");
15         if (mysqli_connect_errno()) {
16             echo("Connessione non effettuata: ".mysqli_connect_error()."<BR>");
17             exit();
18         }
19         $id=$_POST['id'];
20         $sql = "SELECT * FROM users WHERE ID_utente=".$id.";";
21         $ris = $con->query($sql) or die ("Query fallita!");
22         echo "<FORM ACTION='".$_SERVER['PHP_SELF']."' METHOD='post'>";
23         echo "<TABLE><TR><TH>ID utente<TH>Nome utente<TH>Password<TH>Contatore visite";
24         //Ciclo foreach legge gli elementi del resultset $ris
25         foreach($ris as $riga) {
26             echo "<TR><TD>".$riga["ID_utente"];
27             echo "<TD>".$riga["nome_utente"];
28             echo "<TD>".$riga["password"];
29             echo "<TD>".$riga["conta_pres"];
30             echo "<INPUT TYPE='hidden' NAME='id' VALUE=".$riga['ID_utente'].">";
31         }
32         echo "<TR><TD><INPUT TYPE='submit' VALUE='Conferma' NAME='cancella'>";
33         echo "</TABLE></FORM>";
34     }
```

La parte successiva di codice rappresenta la terza fase, quella che provvede alla cancellazione vera e propria del record all'interno della tabella users. Prima di tutto viene effettuata nuovamente la connessione al database (righe 37-40). Viene poi salvato il campo id ricevuto, necessario per individuare il record da cancellare. La riga 44 memorizza la stringa SQL necessaria per cancellare il record

mediante il costrutto DELETE ed esegue la query stessa sul server (riga 45). Viene infine mostrato un messaggio di successo, che consente di richiamare la pagina stessa (riga 47).

```
35        else
36        {
37            $con = new mysqli("localhost","root","","utenti");
38            if (mysqli_connect_errno()) {
39                echo("Connessione non effettuata: ".mysqli_connect_error()."<BR>");
40                exit();
41            }
42            $id=$_POST['id'];
43            //Codice SQL per la cancellazione del record nella tabella
44            $sql = "DELETE FROM users WHERE ID_utente=".$id.";";
45            $ris = $con->query($sql) or die ("Query fallita!");
46            $con->close();
47            echo "<A HREF='".$_SERVER['PHP_SELF']."'>Cancellazione effettuata</A>";
48        }
49    }
50    ?>
```

Nell'esecuzione dello script proponiamo la cancellazione del record con campo ID = 7:

Vengono mostrati i dati del record ricercato:

Facendo clic su Conferma Cancellazione otteniamo l'esecuzione dello script in postback che esegue la stringa SQL di cancellazione. Il messaggio che otteniamo in caso di esito positivo è il seguente:

Puoi trovare il codice di questo esempio nel file Cancella_mysql.php.

METTITI ALLA PROVA

➜ • Applicazione della connessione da php a MySQL • Utilizzo delle sessioni per la persistenza dei dati

Apri il file Elenca_mysql.php

1 Modifica lo script e la struttura della tabella utenti del database users in modo tale da consentire la visualizzazione anche delle immagini che raffigurano i vari utenti. Per fare questo, puoi inserire un semplice campo testo che contiene l'indirizzo del file dell'immagine che raffigura l'utente mostrato a video.

VERIFICA... le competenze

ESERCIZI

Crea gli script che risolvano i problemi proposti.

1 Crea un database MySQL contenente un elenco di nominativi dei rappresentanti di una azienda (nome, cognome, ultimo_fatturato, regione, provincia, provvigione%). Mediante una pagina php fornisci l'elenco aggiornato dei rappresentanti che hanno effettuato l'ultimo fatturato compreso tra due valori immessi dall'utente.

2 Sempre con riferimento all'esercizio precedente, scrivi uno script php per aumentare la provvigione di 2 punti a quei rappresentanti che hanno venduto più di 1000 nella regione Lombardia.

3 Crea un database MySQL in grado di memorizzare i dati per conto di una società di statistica. Il database deve permettere la memorizzazione delle vendite effettuate giornalmente da diversi negozi sparsi in tutta Italia. Per ogni negozio avviene un login, quindi viene memorizzato l'importo del venduto giornaliero. Crea la pagina php di lettura dati dei clienti, accesso da amministratore e accesso da utente (solo per avere un elenco delle sedi dei vari negozi). L'accesso da amministratore deve visualizzare le percentuali di vendite giornaliere per ogni provincia rispetto al totale.

4 Crea un database MySQL per la gestione delle prenotazioni di un insieme di eventi da parte di un utente. Le pagine Web in php sono le seguenti:
- un Form in cui l'utente può inserire il nome dell'evento (una stringa) e il numero di persone per cui effettua la prenotazione. Il Form permette l'inserimento di una prenotazione per volta e l'inserimento di più prenotazioni avviene inviando più volte il Form in una stessa sessione;
- una pagina di raccolta delle prenotazioni che memorizza sul server le prenotazioni fatte col Form del punto precedente. Si assume che se l'utente invia più di una volta il numero di prenotazioni per lo stesso evento nella stessa sessione, il numero di persone viene cumulato;
- una pagina di riepilogo che stampa la lista delle prenotazioni, il totale del numero di persone e l'evento per cui è previsto il massimo numero di prenotazioni. Nel caso ci siano più eventi a cui corrisponde il valore massimo di prenotazioni, si stampa il primo nella lista.

5 Crea un database MySQL per la gestione dell'acquisto di un insieme di prodotti da parte di un utente. Le pagine Web in php sono le seguenti:
- un Form in cui l'utente può inserire il nome del prodotto (una stringa) e il numero di pezzi che vuole acquistare. Il Form permette l'inserimento di un acquisto per volta e l'inserimento di più acquisti avviene inviando più volte il Form in una stessa sessione;
- una pagina di raccolta delle richieste di acquisto che memorizza sul server gli acquisti fatti col Form del punto precedente. Si assume che se l'utente invia più di una volta il numero di pezzi per lo stesso prodotto nella stessa sessione, il numero di pezzi è uguale all'ultimo valore inserito;
- una pagina di riepilogo che stampa la lista dei prodotti, il totale dei pezzi e il prodotto per cui è richiesto il massimo numero di pezzi. Nel caso ci siano più prodotti a cui corrisponde il valore massimo di pezzi, si stampa il primo nella lista.

6 Crea un database MySQL per la gestione della raccolta di preferenze da parte di un utente su film. Le pagine Web in php sono le seguenti:
- un Form in cui l'utente può inserire il titolo del film (una stringa) e la sua valutazione numerica (si può assumere un valore nell'intervallo 1-5, ma non è necessario prevedere il codice per il controllo). Il Form permette l'inserimento di una valutazione per volta e l'inserimento di più valutazioni avviene inviando più volte il Form in una stessa sessione;
- una pagina di raccolta delle valutazioni dei film che memorizza sul server le valutazioni fatte col Form del punto precedente. Si assume che se l'utente invia più di una volta una valutazione per lo stesso film nella stessa sessione, la valutazione è uguale al massimo dei valori inseriti.

ALTERNANZA SCUOLA-LAVORO

Scheda progetto n. 1

GESTIONE FLOTTA AZIENDALE

PROBLEMA

Ti viene chiesto di collaborare con lo staff informatico dell'azienda presso la quale stai svolgendo il periodo di alternanza per realizzare un sito Web dinamico per la gestione online di un parco automezzi aziendale. L'azienda possiede alcune vetture, autocarri, furgoni ecc. che vengono concessi ai propri dipendenti secondo le seguenti fasce di utilizzo:
- **uso esclusivamente personale (FASCIA A);**
- **uso promiscuo, cioè per un utilizzo in parte per necessità legate all'azienda e in parte per uso personale (FASCIA B);**
- **uso esclusivamente aziendale (FASCIA C).**

Gli automezzi possono essere assegnati dall'amministratore mediante due modalità:
- **a noleggio con pagamento al km (pool);**
- **in uso gratuito con rimborso chilometrico a carico dell'azienda (benefit).**

L'interfaccia della pagina Web deve essere una sorta di "cruscotto elettronico" nel quale l'amministratore può decidere l'operazione da svolgere.

Prerequisiti e obiettivi formativi

- Saper progettare un database.
- Conoscere il linguaggio SQL.
- Conoscere il linguaggio php.
- Saper interfacciare le pagine php con i database.

Indicazioni per la progettazione

Il sito Web, di tipo dinamico, deve mostrare due diverse interfacce: una per l'utente (dipendente) e una per l'amministratore per consentire:
- la gestione anagrafica degli automezzi;
- il monitoraggio delle percorrenze e dei rimborsi;
- la valutazione delle richieste dei dipendenti per effettuarne l'assegnazione;
- la consultazione dell'elenco di automezzi disponibili, a partire dal periodo prescelto;
- la richiesta dell'automezzo in concessione;
- la cancellazione o l'aggiornamento del noleggio richiesto.

Traccia per la realizzazione

Il database deve essere creato prevedendo anche una tabella (Users) che contenga i dati degli utenti e degli amministratori, necessari per l'autenticazione da effettuarsi sul sito. Nella creazione delle altre tabelle è necessario tenere presente quanto enunciato nella traccia del problema, pertanto sono necessarie le tabelle Dipendenti, Amministratori, Automezzi, oltre alle tabelle ponte utili per gestire le relazioni. Le pagine php devono prima di tutto consentire l'autenticazione mediante una pagina in postback che confronti i dati di login (utente e password) con quelli presenti nella tabella Users. Utilizza le variabili di sessione per garantire l'accesso sicuro alla pagina dell'utente dipendente e dell'utente amministratore.

Preparazione e presentazione dei risultati

La soluzione di questo problema prevede la creazione delle tabelle del database, con i vincoli relativi e la creazione delle pagine Web in php. È necessario che venga effettuata una simulazione completa con un campione di ogni pagina per verificare gli accessi e la distribuzione dei risultati. In collaborazione con il tuo tutor aziendale, riporta in un documento tutte le osservazioni e le proposte di modifica e/o miglioria del progetto stesso.

ALTERNANZA SCUOLA-LAVORO

Proposta operativa n. 1

STUDIO CONSULENZE TRIBUTARIE

PROBLEMA

Molti studi di consulenza tributaria ricevono reclami da parte dei propri clienti quando comunicano la parcella, al termine del lavoro svolto. Con l'obiettivo di giustificare il contributo economico richiesto al cliente, si stabilisce di predisporre un prospetto dettagliato indicante in modo sistematico ogni attività svolta per il cliente e il relativo tempo impiegato. Per fare questo, è necessario realizzare un sito Web dinamico, operante in intranet all'interno dell'ufficio, che permetta a ogni impiegato (operatore) di caricare il tipo di attività svolta e il tempo dedicato per ciascun cliente e ciascuna pratica.

Passi operativi

1 Realizza un piccolo database.
2 Discuti e analizza insieme ai responsabili dell'ufficio come realizzare le interfacce utente del sito Web dinamico.
3 Realizza le pagine Web in php.

Le fasi richiedono l'esecuzione di attività che potrebbero coinvolgere altri dipendenti della organizzazione: è quindi necessario coordinare gli interventi, soprattutto quelli relativi al punto **2**, in modo da arrecare il minor disturbo agli operatori.

Proposta operativa n. 2

GESTIONE ABBINAMENTI AZIENDE-ALUNNI ALTERNANZA

PROBLEMA

Si vogliono gestire i dati relativi all'Alternanza Scuola-Lavoro degli allievi del tuo istituto. Per fare questo ti viene chiesto di realizzare un sito Web dinamico in grado di gestire le aziende, i docenti tutor e gli allievi, suddivisi per classe e per indirizzo. Al sito possono accedere i docenti tutor e le aziende, che vengono identificati dopo una fase di autenticazione. Le aziende accedono per scaricare e stampare la documentazione relativa ai contratti di collaborazione e per conoscere i dati degli alunni che svolgeranno l'attività di tutoraggio presso di loro, mentre i docenti tutor possono stampare le convenzioni con le aziende e inserire i dati della valutazione aziendale al termine dell'alternanza.

Passi operativi

1 Realizza dapprima un database contenente i dati utili alla memorizzazione dei dati, strutturandolo da una parte con i dati delle aziende convenzionate che hanno dato la propria adesione, dall'altra con quelli degli allievi e con gli abbinamenti allievo-azienda-docente tutor.
2 Realizza quindi il sito Web, di tipo dinamico, che deve mostrare due diverse interfacce: una per l'azienda e l'altra per i docenti tutor. L'azienda immette: dati anagrafici, referente aziendale, responsabile, richiesta studenti (numero, specializzazione e date di disponibilità). Il docente invece immette i dati della valutazione aziendale per ciascun allievo e stampa il contratto e la convenzione con l'azienda.

CLIL

Key concepts

Server-side programming with php

php is an interpreted server-side language. Its syntax derives from that of the C language, even though it employs peculiar technologies for managing http-based communication.
The names of variables must start with the dollar sign ($).

It is a weakly typed language: in fact, variables need not be declared explicitly, and their data type is automatically and implicitly assigned by the language itself when the variables are first initialized.

The end result of the interpretation of a php page is HTML code.

With php, http dialog is achieved with the associative arrays $_POST and $_GET.

Using the postback technique, a php page can call itself to perform a recursive http dialog.

Http dialog is maintained by means of cookies, sessions, files and database connections.

php can connect with Access by means of the ADODB library or by means of MySQL using the mysqli object.

Quiz

Multiple choice

1 An http response contains:
- **a** the php code of the request
- **b** the content of only the POST fields transmitted by the user’s Form
- **c** the content of the fields transmitted by the user’s Form
- **d** the content of only the GET fields transmitted by the user’s Form

Unscramble

1 Reorder the following lines of code so that they can display on screen three names: Paolo, Riccardo e Zeev (bold, italic, underlined):

- **a** echo “Zeev”; (____)
- **b** echo “</G>”; (____)
- **c** ?> (____)
- **d** echo “<I>”; (____)
- **e** echo “Riccardo”; (____)
- **f** echo”<U>”; (____)
- **g** echo “Paolo”; (____)
- **h** echo “<G>”; (____)
- **i** <?php (____)
- **j** echo “</I>”; (____)

True/False

1 The type Boolean is actually an integer that can be either zero (false), or non-zero (true). V F
2 The function is_bool() checks if a variable contains 0 or 1. V F
3 Casting between integer and float is not possible. V F
4 php does not support Unicode strings. V F

UNITÀ 5

La prova scritta di Informatica

LEZIONE 1
La nuova prova scritta di Informatica (“la buona scuola”): note generali

LEZIONE 2
Car pooling (ITIA Ordinaria 2017)

LEZIONE 3
Eventi dal vivo (ITIA Ordinaria 2015)

LEZIONE 4
Trasporto passeggeri (City2City) (ITSI Ordinaria 2016)

LEZIONE 5
Abbigliamento (Gamma) (ITSI Esempio 2016)

CONOSCENZE

- Conoscere la strutturazione delle prove scritte di Informatica
- Avere una panoramica della tipologia di richieste

COMPETENZE

- Saper affrontare la prova scritta di Informatica
- Essere in grado di impostare la soluzione di un progetto di medie dimensioni
- Sapersi approcciare a una prova scritta a tempo

ABILITÀ

- Saper organizzare un progetto in base al tempo a disposizione
- Essere in grado di valutare priorità ed essenzialità di una richiesta progettuale
- Progettare e realizzare applicazioni informatiche con basi di dati

I codici delle soluzioni dei temi di maturità descritti nella UDA e di altri particolarmente significativi sono disponibili nel CD-ROM nella cartella SOLUZIONI MATURITA: è sufficiente copiarne il contenuto nella cartella locale di XAMP, cioè in C:\xamp\htdocs, creando la cartella maturita.

La nuova prova scritta di Informatica ("la buona scuola"): note generali

IN QUESTA LEZIONE IMPAREREMO...

- come è strutturata la prova d'esame
- come impostare la soluzione della traccia ministeriale

Pag. 1/2

A

Sessione ordinaria 2016
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

M967 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITSI - AMMINISTRAZIONE, FINANZA E MARKETING
ARTICOLAZIONE "SISTEMI INFORMATIVI AZIENDALI"

Tema di: INFORMATICA - ***Tipologia b***

Il candidato (che potrà eventualmente avvalersi delle conoscenze e competenze maturate attraverso esperienze di alternanza scuola-lavoro, stage o formazione in azienda) svolga la prima parte della prova e due tra i quesiti proposti nella seconda parte.

B

PRIMA PARTE

La compagnia City2City è una società italiana di recente costituzione che offre collegamenti passeggeri diretti verso varie città europee. City2City si vuole inserire dinamicamente sul mercato del trasporto passeggeri, offrendo un servizio efficiente e a costi contenuti. Il suo target sono tutti quei soggetti che non amano spostarsi in aereo, non hanno la possibilità di usare un mezzo proprio e desiderano viaggiare comodamente, trasportando facilmente il proprio bagaglio.

La compagnia, almeno per la fase iniziale di avvio delle attività, ha scelto di offrire collegamenti di andata e di ritorno che partono dalla propria sede operativa, situata in una città di medie dimensioni del centro Italia, e sono diretti ad alcune città europee, raggiungibili in 24 ore di viaggio al massimo.

Si è quindi dotata di un parco di autobus GT ("Gran Turismo") forniti di differenti livelli di *comfort*. La manutenzione degli autobus GT è affidata ad una serie di ditte esterne, e deve essere comunque tenuta sotto attento controllo dalla City2City. Puntando sulla qualità e sulla sicurezza del servizio offerto, si è scelto che il personale viaggiante sia costituito da un autista principale ed un secondo autista che si alternano alla guida, oltre che da un assistente di viaggio in grado di offrire servizi di cortesia ai passeggeri (es. riviste, frigo bar, noleggio tablet, noleggio videogiochi o film) di cui la City2City periodicamente rifornisce i suoi mezzi.

La compagnia è interessata a fidelizzare i clienti e prevede quindi sistemi di registrazione degli utenti, con tessera virtuale a punti e indagini di mercato di *customer satisfaction*. Infine, essa intende offrire sistemi di prenotazione on-line.

Il candidato, fatte le opportune ipotesi aggiuntive:

C

1. identifichi le principali aree del sistema informativo della compagnia City2City e le soluzioni tecnologiche necessarie alla sua implementazione;
2. concentrandosi poi sulla porzione del sistema informativo che gestisce l'attività di trasporto, sviluppi uno schema concettuale della relativa base di dati, che dovrà prevedere:
   - i collegamenti offerti, registrando per ciascuno la città collegata, il giorno della settimana, l'orario previsto di partenza e di arrivo, se il collegamento è di andata o di ritorno
   - il personale viaggiante, del quale oltre ai dati anagrafici interessa il ruolo ricoperto come sopra descritto
   - il parco di autobus GT, con i dati identificativi di ciascun mezzo, le relative caratteristiche e i dati necessari a gestirne la manutenzione
   - i viaggi effettuati, registrando per ciascuno la data in cui è stato svolto, l'orario effettivo di partenza e di arrivo, il numero dei passeggeri effettivi, l'autobus GT utilizzato e il personale viaggiante impiegato;

Pag. 2/2

A

Sessione ordinaria 2016
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

M967 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITSI - AMMINISTRAZIONE, FINANZA E MARKETING
ARTICOLAZIONE "SISTEMI INFORMATIVI AZIENDALI"

Tema di: INFORMATICA - ***Tipologia b***

C

3. derivi il corrispondente schema logico relazionale;
4. sviluppi in linguaggio SQL le query per ottenere le seguenti informazioni:
   a) elenco dei viaggi di andata svolti verso una determinata città, in un intervallo di date fornite in input;
   b) per ciascuna città collegata, calcolare la media del tempo di percorrenza dei viaggi di andata effettuati nel mese di agosto dell'anno corrente.

D

SECONDA PARTE

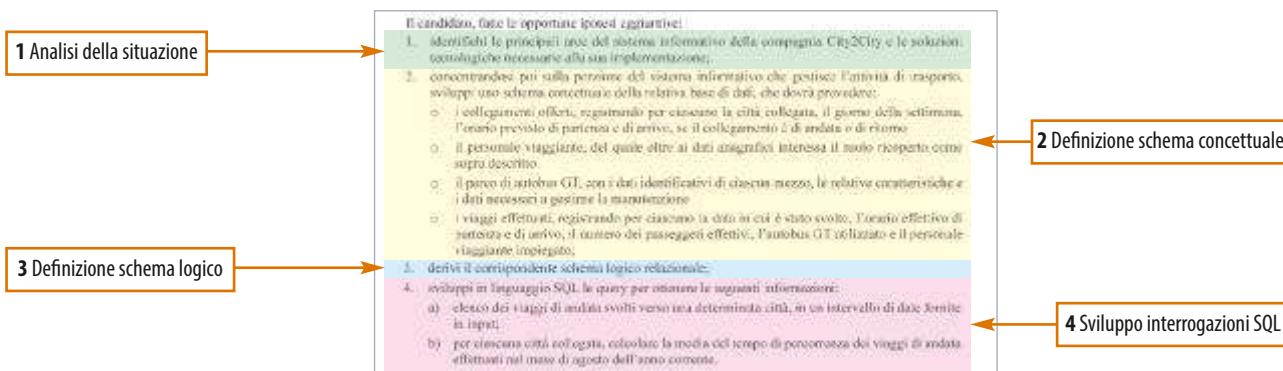
1. In relazione al tema proposto nella prima parte, si supponga che la compagnia "City2City" desideri attivare con i propri clienti operazioni di marketing non convenzionale, che tengano conto di fattori come stile di vita, interessi, realtà famigliare, eccetera.
   Il candidato a tale scopo:
   a. integri la base di dati sviluppata nella prima parte inserendo opportunamente un'entità "Cliente";
   b. ne definisca gli attributi utili alle operazioni di marketing ipotizzate, motivando le scelte fatte, e ne derivi lo schema logico;
   c. infine sviluppi le pagine web necessarie alla registrazione di un nuovo cliente, in un linguaggio di programmazione a propria scelta.
2. Il candidato, dopo aver illustrato le motivazioni che spingono un'azienda di medio-piccole dimensioni ad utilizzare sistemi in rete, esponga le problematiche relative alla progettazione e realizzazione delle infrastrutture hardware e della dotazione software per i servizi che si intendono attivare, anche con riferimento all'impatto sotto il profilo organizzativo.
3. Il candidato illustri la filosofia progettuale ed i principali moduli dei sistemi ERP, approfondendo in particolare un modulo di sua conoscenza, anche evidenziandone le funzionalità, il campo di applicazione, l'integrazione con gli altri moduli.
4. Nel Sistema Sanitario Nazionale è stata di recente attuata la dematerializzazione delle prescrizioni di farmaci ai pazienti. Il sistema prevede che il medico, anziché produrre una ricetta cartacea con i farmaci prescritti al paziente, registri la prescrizione su un portale dedicato, inclusa la segnalazione di situazioni che comportino l'esenzione dal ticket, e consegni al paziente un promemoria contenente anche il codice della prescrizione. Con tale codice il paziente può ritirare il farmaco presso qualsiasi farmacia. A partire da tale esemplificazione, il candidato illustri come le tecnologie informatiche stanno cambiando il rapporto tra cittadini e Pubblica Amministrazione e l'efficienza dei servizi da essa forniti; discuta quindi gli aspetti di riservatezza nel trattamento dei dati, compresi quelli sensibili, e i sistemi oggi disponibili per garantire la sicurezza della loro trasmissione e conservazione.

Durata massima della prova: 6 ore.
È consentito l'uso di manuali tecnici e di calcolatrici tascabili non programmabili.
È consentito l'uso del dizionario bilingue (italiano-lingua del paese di provenienza) per i candidati di madrelingua non italiana.
Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla dettatura del tema.

A Intestazione del documento | B Breve presentazione della situazione | C Elenco delle richieste | D Quesiti teorici

Generalità

La recente riforma della scuola superiore ha introdotto delle novità anche nella seconda prova scritta di Informatica: alla parte classica, consolidata da ormai un decennio – che consiste nello studio di una situazione reale per arrivare alla progettazione del database che soddisfi un insieme di interrogazioni (di solito 4 query in linguaggio SQL) – è stata aggiunta una parte teorica con la richiesta di rispondere a due quesiti a scelta tra i quattro proposti.

Le lezioni che seguono propongono i testi completi di alcune delle ultime tracce ministeriali fornendo la soluzione della parte di progettazione del database, mentre per le risposte ai quesiti teorici viene fornita una sintesi rimandando alle UDA del volume dove questi contenuti sono stati discussi.

Prima parte della prova

Per quanto riguarda la **PRIMA PARTE**, in generale le richieste della traccia sono molto dettagliate e articolate e richiederebbero, per una soluzione completa da parte dello studente, un tempo molto superiore alle sei ore messe a disposizione.
Dopo l'intestazione della pagina (figura di pag. 350, riquadro A), viene presentata una situazione (figura di pag. 350, riquadro B) in cui generalmente il tema proposto, pur trattando argomenti che dovrebbero essere noti al candidato, è più adatto a un progetto di ampio respiro da sviluppare in classe/laboratorio in un periodo di tempo piuttosto lungo che non per un tema di maturità.
Le richieste (figura di pag. 350, riquadro C) spaziano da un'analisi approfondita delle tematiche legate alla sicurezza e alla riservatezza delle informazioni a un progetto completo di database in cui le modalità di salvaguardia delle informazioni nel tempo sono lasciate a carico del candidato: dette richieste sono riportate in una apposita sezione introdotta dalla frase «Il candidato, fatte le opportune ipotesi aggiuntive:». Sono generalmente suddivise in quattro (a volte anche sei o più) punti:
– analisi del problema proposto;
– realizzazione dello schema concettuale del database;
– progettazione dello schema logico del database;
– realizzazione di alcune interrogazioni in linguaggio SQL.

Progettazione della base di dati

I primi tre punti riguardano sostanzialmente la progettazione della base di dati, che richiede sempre le seguenti fasi ben definite:
– analisi della situazione descritta;
– formulazione di eventuali ipotesi aggiuntive per meglio definire l'ambito del progetto;
– progettazione di uno schema concettuale dei dati, per esempio mediante i diagrammi E-R;
– definizione dello schema logico con l'analisi delle tabelle per studiarne la normalizzazione.

Realizzazione delle interrogazioni in linguaggio SQL

Il quarto punto richiede invece di realizzare diverse query (da quattro a sei), generalmente in ordine crescente di difficoltà, che comportano la conoscenza approfondita delle possibilità offerte dal linguaggio SQL.

Seconda parte della prova

Nella **SECONDA PARTE**, introdotta da "la buona scuola", viene richiesto di rispondere a due quesiti tra i quattro proposti: tali quesiti spaziano tra domande teoriche, query aggiuntive oppure veri e propri "ulteriori progetti" da sviluppare, mantenendo immutato il tempo assegnato (!).

Schema generale di soluzione

Le tracce delle prove scritte sono sempre strutturate sul modello appena descritto e quindi si può proporre il seguente schema di risoluzione, facilmente adattabile a ogni situazione:
– analisi;
– formulazione delle ipotesi aggiuntive;
– realizzazione dello schema concettuale;
– realizzazione dello schema logico;
– interrogazioni.

Analisi

Non entreremo nei dettagli di come si effettua l'analisi, dato che nel corso di studi più volte l'argomento è stato trattato: ci limiteremo a dare alcuni suggerimenti, primo fra tutti quello di leggere più volte e attentamente il testo prima di decidere il modello dei dati.

Non bisogna avere fretta di avviare il progetto: è bene leggere dapprima tutto il testo e successivamente rileggere più volte la parte B, dove viene presentata la situazione, in modo da avere ben chiaro "cosa si deve fare".

Nella rilettura è utile iniziare a riempire la tabella del glossario dei termini, fornendo per ciascun termine una breve descrizione e indicando a quali altri termini è collegato.

ESEMPIO

Glossario dei termini Ambulatorio Medico
La situazione prevede la definizione di un ambulatorio medico dove i pazienti sono visitati, viene stilato un referto e definita una diagnosi sulla base dei sintomi dichiarati dai malati, eventualmente individuando (o essendo a conoscenza di) una patologia della quale soffre il paziente, e può avere il seguente glossario dei termini.

Termine	Descrizione	Collegamenti
Paziente	Ogni persona che usufruisce dei servizi dell'ambulatorio	Visita, Referto, Medico
Medico dell'ambulatorio	Medico che gestisce l'ambulatorio e visita i pazienti	Visita, Referto, Paziente
Visita	Controllo dello stato di un paziente con redazione di un referto	Medico, Paziente, Referto
Referto	Nota stilata dal medico al termine della visita, contenente sintomi riscontrati, patologie diagnosticate, medicinali somministrati	Paziente, Medico, Visita, Sintomo, Patologia, Medicinale
Patologia	Malattia di cui è affetto il paziente: nota a priori o diagnosticata nel referto. Comprende anche le allergie	Paziente, Referto
Sintomo	???	???
Medicinale	Farmaco somministrabile a seguito della visita	Referto, Patologia

L'introduzione di un glossario facilita la successiva definizione delle entità e delle relazioni tra di esse: nei casi in cui un termine non fosse noto o non fosse chiaro il suo significato (nel nostro esempio indicato con ???), lo studente dovrà automaticamente effettuare un "supplemento di analisi" per meglio definire ogni componente della tabella prima di procedere alla progettazione del database.

Ipotesi aggiuntive

Generalmente, anche per ragioni di spazio e di tempo, la situazione da studiare è "una parte" di una situazione più complessa: inoltre, può capitare che il dominio di appartenenza non sia completamente noto al candidato.

Altre volte, la situazione proposta implica l'interfacciamento con sistemi già preesistenti, sia hardware che software: per tutto quanto non completamente specificato, è necessario porre delle ipotesi aggiuntive al testo, in modo da "sciogliere ogni perplessità e/o ambiguità".

Dato che le ipotesi aggiuntive possono essere formulate "a piacere", si consiglia di scegliere sempre quelle che "rendono la vita più facile", cioè di porre condizioni e/o ipotesi tali che semplifichino il più possibile la situazione, senza tuttavia "sminuire e/o alterare la traccia ministeriale".

Schema concettuale

Nella definizione dello schema E-R si deve sempre procedere per affinamenti successivi, tenendo presenti alcuni consigli:
- non bisogna inserire come entità "l'ambito del problema": per esempio, nel caso precedente, l'ambulatorio non è un'entità;
- individuare quali sono le entità statiche e quali quelle dinamiche, ricordandosi per queste ultime di indicare l'attributo TIMESTAMP, cioè per esempio la data registrazione e, dove necessario, anche l'attributo ora;
- nel dubbio, introdurre chiavi primarie artificiali autoincrement, soprattutto per le entità di tipo dinamico, che possono anche essere utilizzate come "progressivo registrazione";
- quando si è completato il modello E-R, rileggere le interrogazioni richieste, verificare se ogni dato può essere reperito dalle entità e indicarlo come attributo sull'entità stessa;
- tutti gli attributi descrittivi di catalogazione che si ripetono (per esempio la tipologia, la categoria, il settore, il genere, la causale ecc.) devono essere isolati subito in una entità (tabella dizionario), senza aspettare di individuarli successivamente durante la fase di normalizzazione;
- le tabelle di tipo dizionario possono essere composte anche da un solo attributo ma, generalmente, vengono indicate con due campi, un codice progressivo e un campo descrittivo (per esempio: ID_Tipo(pk), descrizione).

È importante ricordarsi di motivare sempre le scelte fatte nel caso di ambiguità, evidenziando le ipotesi aggiuntive e gli eventuali vincoli introdotti.

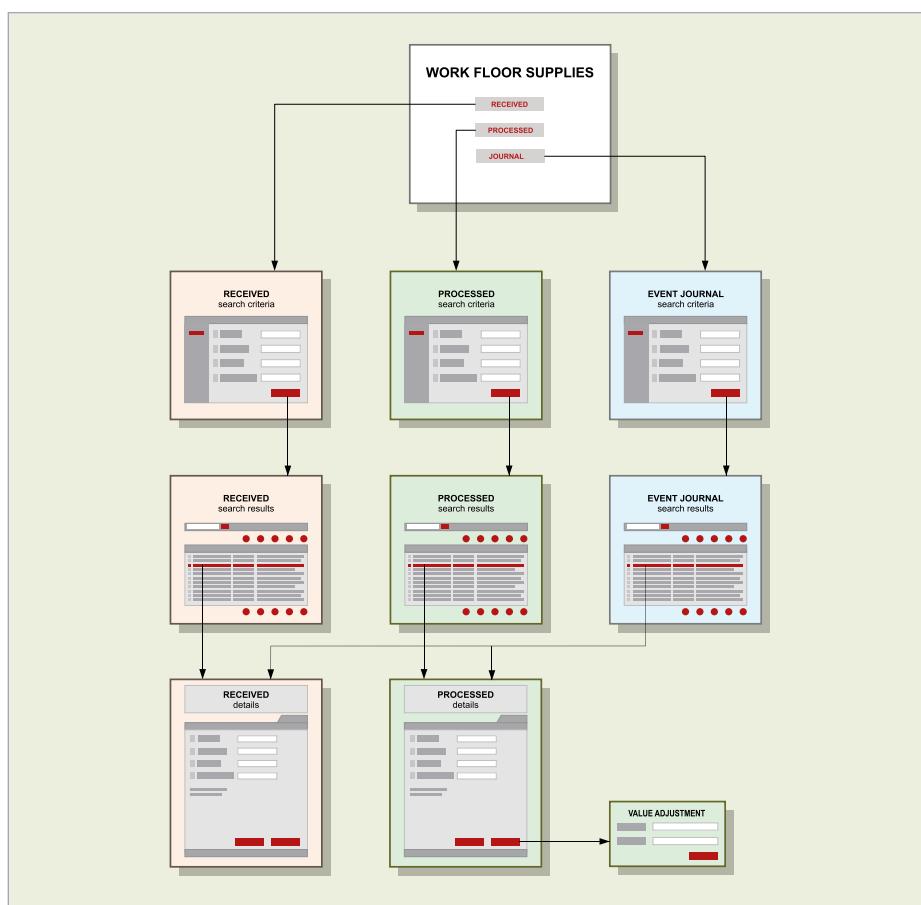
Schema logico

Anche per lo schema logico riportiamo alcuni semplici consigli.
- Indicare per tutti i campi significativi una descrizione, soprattutto per quelli calcolati e autoincrement.
- Definire come tipo stringa tutti i campi numerici sui quali non vengono eseguite operazioni, come il CAP, il nr. di telefono, la partita IVA ecc.
- Utilizzare dei "dati di prova" popolando le tabelle, in modo da verificare la completezza della soluzione proposta: tali dati saranno utilizzati per verificare in seguito le query.
- Analizzare ogni singola tabella per individuare eventuali violazioni delle forme normali, anche se non richiesto esplicitamente nella traccia.

Interrogazioni

Di seguito sono riportati alcuni suggerimenti per la realizzazione delle query.

- Iniziare a scrivere le query seguendo l'ordine delle richieste, dato che solitamente hanno complessità crescente: se ci fossero problemi nella loro realizzazione, individuarne la causa provando a modificare le tabelle prima definite.
- Quando possibile, cercare di scrivere query che coinvolgono il minor numero di tabelle.
- Gli errori classici vengono commessi omettendo la clausola di raggruppamento GROUP BY quando si utilizzano funzioni di aggregazione (COUNT, SUM, AVG ecc.), oppure quando è presente HAVING (HAVING pone condizioni sui gruppi e quindi deve sempre essere accompagnato da GROUP BY, perché HAVING è sempre "assieme" a una funzione di aggregazione).
- Vicino a HAVING non ci può essere una condizione sulle singole righe: questa deve essere messa vicino a WHERE.
- Se si devono scrivere query complesse annidate, si consiglia di utilizzare la clausola AS (alias) per assegnare un nome a un'espressione di calcolo o al valore restituito da una funzione di aggregazione.

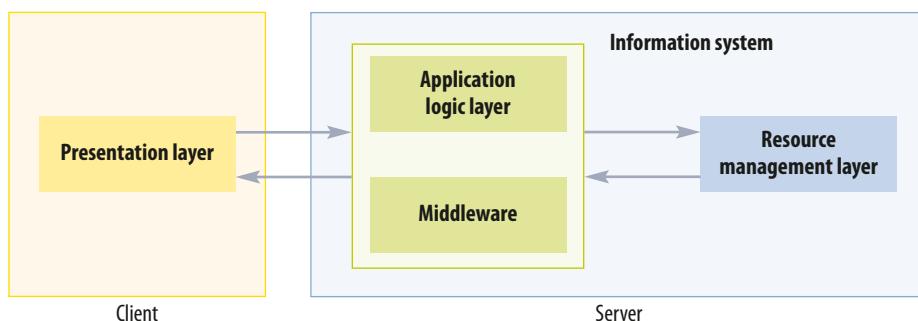
Moduli software

Oltre alla progettazione del database, tra le richieste è spesso presente la pubblicazione di una o più pagine Web: prima di passare alla loro realizzazione è necessario "riorganizzare le idee", definendo una scomposizione in moduli funzionali.
Risulta efficace disegnare un diagramma che riporta ogni modulo, evidenziando le interfacce tra i moduli stessi: nel caso di pagine Web, è opportuno indicare lo **storyboard**, oppure lo schema delle connessioni ipertestuali, come riportato nell'esempio seguente dove, da un menu generale, si accede a tre pagine.

Storyboard
Letteralmente, dall'inglese "tavola (*board*) della storia (*story*, intesa come racconto)"; il termine storyboard viene generalmente utilizzato per indicare la rappresentazione grafica, sotto forma di sequenze disegnate in ordine cronologico, delle inquadrature di un fumetto o di un'opera filmata, oppure di una presentazione.

In ogni pagina deve essere presente una barra di navigazione che permette di accedere direttamente alle altre pagine con un solo passaggio.

Se non vi sono altre indicazioni, si procede quindi alla realizzazione del codice di un modulo a piacere, individuando quello più significativo che allo stesso tempo presenta "meno insidie".

Codice lato server

Nelle tracce degli ultimi anni è sempre presente la richiesta di connettere tramite Internet pagine statiche a database: è quindi necessario l'utilizzo di un linguaggio di programmazione dinamico lato server (php, oppure ASP).

Three-tier
Il modello three-tier si struttura, come indica il nome stesso, su tre livelli: il primo livello, che è quello dei client che gestiscono l'interfaccia con l'utente, quello intermedio, middle-tier, su cui deve appoggiarsi tutta la logica di analisi delle richieste dei client per ottimizzare l'accesso al terzo livello, che è quello che si limita a fornire i dati dinamici che verranno usati dalla logica implementata nel middle-tier per eseguire le operazioni richieste dai client.

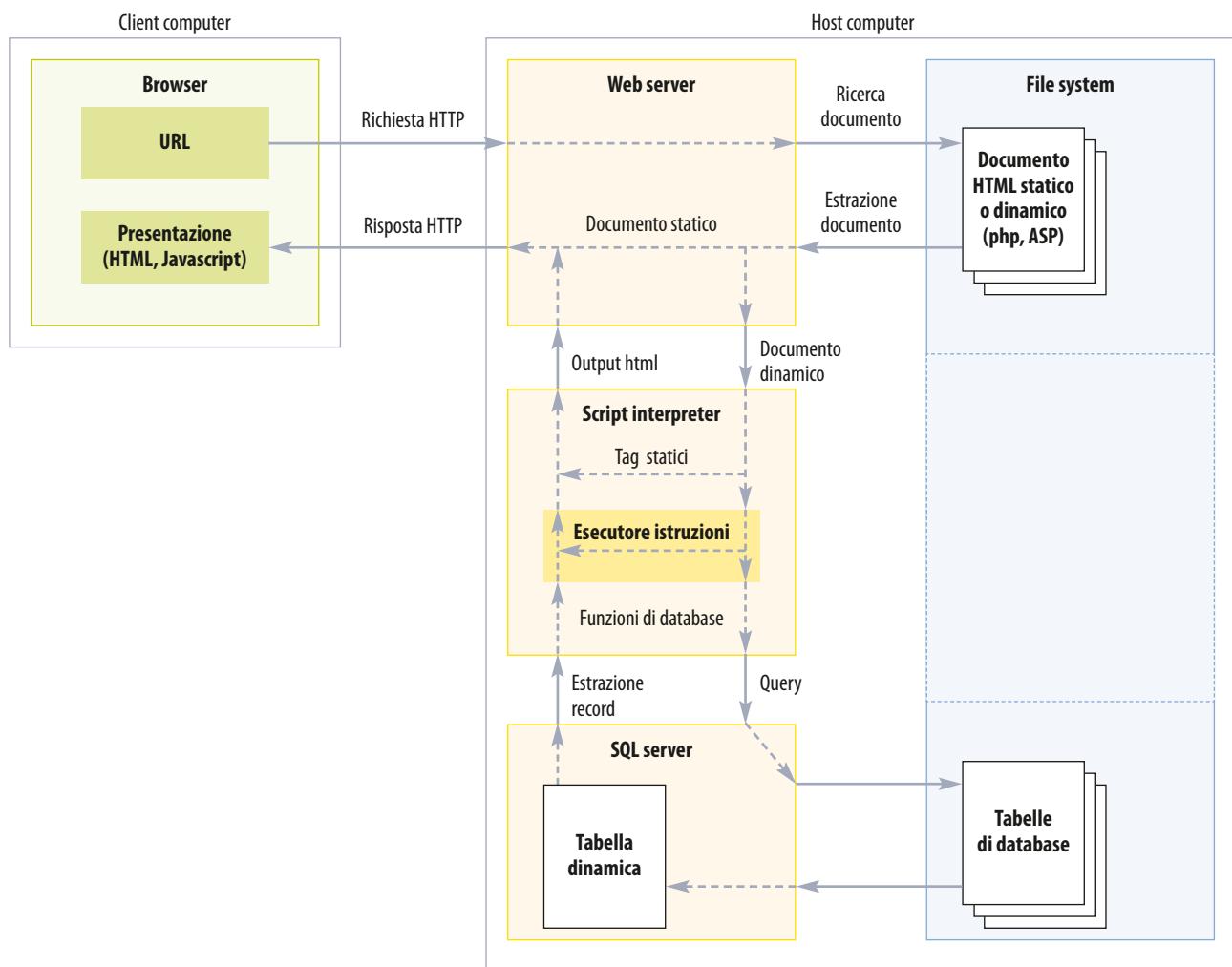
Per ciascuna pagina da scrivere è doveroso predisporre uno schema **three-tier**:

a interfaccia lato client (presentation layer);
b moduli per la generazione di contenuti dinamici;
c connessione al database.

Il tempo a disposizione non permette di realizzare completamente le pagine richieste e quindi è necessario introdurre delle semplificazioni mediante pseudocodifica. Lo pseudocodice dello schema generale di una pagina dinamica è il seguente:

```
<intestazione pagina>
<connessione alla banca dati>
<query>
...
<estrazione riga>
<presentazione dati dinamici>
...
<barra di navigazione>
```

Se non si riesce a scrivere lo script completo, si consiglia "almeno "di codificare in linguaggio di programmazione il meccanismo di connessione al database, il segmento che effettua la query SQL e una parte del segmento di presentazione.

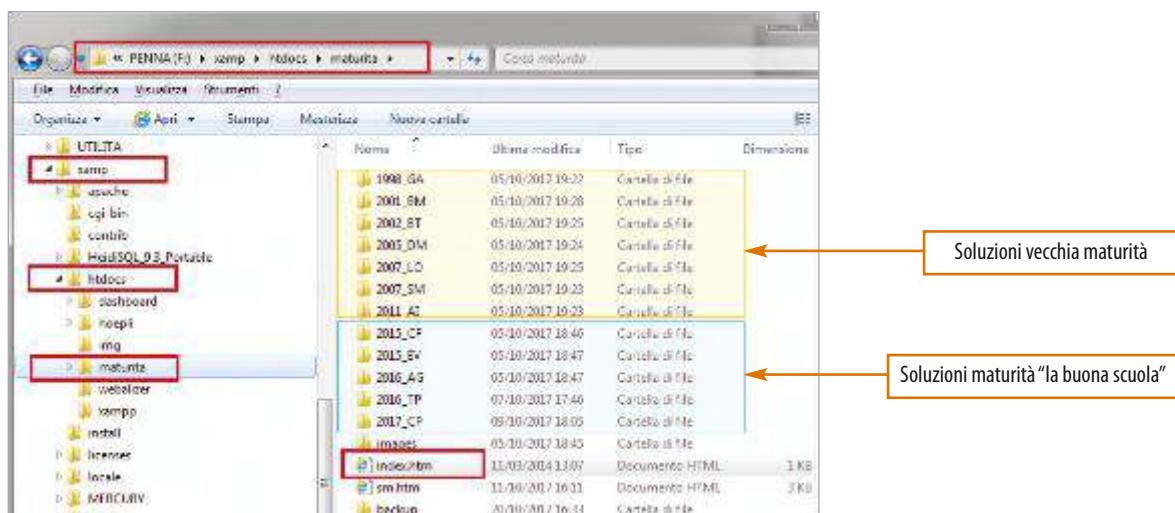
Descrizione della piattaforma operativa

Anche se non viene esplicitamente richiesto, è opportuno riportare al termine dell'elaborato una breve descrizione della piattaforma operativa: a pagina seguente proponiamo uno schema che può essere utilizzato in ogni situazione.

Una applicazione Web si basa sull'impiego di protocolli di comunicazione standard che utilizzano la rete Internet: il computer dell'utente (client) ha in esecuzione un browser (Internet Explorer, Chrome, Netscape ecc.), che viene quindi definito "client", dove viene inserito l'indirizzo di una pagina Web (URL).
In questo modo, il browser chiede la pagina all'host specificato (RICHIESTA HTTP), nel quale deve essere in esecuzione un Web server (Apache, IIS ...) che riceve le richieste dal client e le processa. La pagina di risposta può essere statica, cioè consistere in un unico documento HTML, che il server inoltra al client (RISPOSTA HTTP) allegando eventuali files multimediali presenti nella pagina (immagini, applet, suoni ecc.), oppure dinamica, e in tale situazione il Web server inoltra la pagina (DOCUMENTO DINAMICO) a un altro programma, lo "script interpreter", che legge e interpreta i tag contenuti nel documento, mandando direttamente in uscita verso il Web server i tag HTML senza modificarli mentre esegue le istruzioni contenute all'interno dei tag di script del linguaggio php, ASP ecc.
Le uscite prodotte dalla elaborazione, che normalmente sono in forma di tag HTML (OUTPUT HTML), tornano verso il Web server, che si vede ritornare la pagina trasformata da programma a risultato della elaborazione del programma (cioè una normale pagina HTML) da inoltrare al client. È compito del client trasformare i tag contenuti nell'OUTPUT HTML nella finestra del browser (PRESENTAZIONE), eventualmente elaborando gli eventuali tag di script client side (JAVASCRIPT).

Tra i tag presenti nel linguaggio lato server sono comprese le istruzioni di consultazione ed elaborazione delle tabelle presenti nella base di dati che risiede sul server, con i relativi programmi che permettono la sua gestione (SQL server).

L'ambiente operativo è costituito da tre componenti lato server (oltre al client, che però è un browser standard): Web server, script interpreter, SQL server. Nell'ipotesi di operare su una piattaforma "Open source" (software sviluppato in progetti aperti di cui sono gratuitamente disponibili sia gli eseguibili che i sorgenti), i tre componenti possono essere rispettivamente: Apache, php, MySQL. Tutti questi componenti operano sia sulla piattaforma linux che sulla piattaforma win 32/64.

XAMP

L'ambiente operativo più utilizzato per la realizzazione di applicazioni Web è la piattaforma XAMP, composta da:
– X: sistema operativo Windows o Linux;
– A: Apache come server Web;
– M: MySQL come server di database;
– P: php come linguaggio di script lato server.

Ultimamente è stato aggiunto il linguaggio Perl, quindi l'acronimo è divenuto XAMPP.
È scaricabile gratuitamente all'indirizzo www.apachefriends.org/it ed è "auto-configurante".

Ambiente di sviluppo

Come ambiente di sviluppo basta un semplice editor di testi, oppure uno strumento di sviluppo come Scite (*SCIntilla based Text Editor*), scaricabile gratuitamente dal sito www.scintilla.org/SciTE o Dreamweaver, programma per la realizzazione di siti Web prodotto da Macromedia (a pagamento).
L'architettura hardware non viene sviluppata, perché generalmente nel tema di informatica non è richiesta dalle specifiche.
I codici delle soluzioni dei temi di maturità proposti nelle successive lezioni sono disponibili nel CD-ROM, nella cartella SOLUZIONI MATURITA: è sufficiente copiarne il contenuto nella cartella locale di XAMP, cioè in C:\xamp\htdocs creando la cartella maturita e caricando il menu presente nel file index.htm.

Elenco storico delle tracce ministeriali

Riportiamo per completezza l'elenco delle tracce proposte nei diversi indirizzi con seconda prova INFORMATICA: le tracce evidenziate in giallo sono quelle di cui viene fornita la soluzione completa: i relativi PDF sono disponibili nella cartella MATERIALI / sottocartella TRACCE MATURITA del CD-ROM allegato al volume.

Legenda
Testo rosso: ITIA- Mercurio • Testo azzurro: ITSI- Abacus • Testo verde: Informatica gestionale • Testo evidenziato in giallo: prove di cui sono presenti le soluzioni su CD-ROM • Testo con fondino grigio: prove di "la buona scuola"

Anno	Traccia	Descrizione
2017	Informatica (ITSI) Ordinaria	Car pooling
	Informatica (ITSI) Suppletiva	Gestione assenze
2016	Informatica (ITIA) Ordinaria	Trasporto passeggeri (City2City)
	Informatica (ITIA) Suppletiva	Centri di impiego
	Informatica (ITIA) Straordinaria	Comuni documentazione ISEE
	Informatica (ITIA) Esempio	Abbigliamento (Gamma)
2015	Informatica (ITIA) Ordinaria	Eventi dal vivo
	Informatica (ITIA) Suppletiva	Risorse multimediali
	Informatica (ITIA) Esempio	Consorzio formaggio
	Informatica (Abacus-Sirio)	Stabilimenti balneari
2014	Informatica (Abacus/Ordinamento)	Alternanza scuola-lavoro
2013	Informatica (Abacus/Ordinamento)	Dogana aeroporto
	Informatica (Mercurio)	Biglietteria museo on-line
2012	Informatica (Ordinamento)	Gestione immobili ASL
2011	Informatica (Abacus/Ordinamento)	Parchi regionali
	Informatica (Mercurio)	Agenzia immobiliare
2010	Informatica (Ordinamento)	Società di riparazione
2009	Informatica (Abacus/Ordinamento)	Società telefonica
	Informatica (Mercurio)	Pubblicazione riviste
2008	Informatica (Ordinamento)	Olimpiadi informatica
2007	Informatica (Abacus/Ordinamento)	e-book shop
	Informatica (Mercurio)	Studio medico
2006	Informatica (Ordinamento)	E-learning
2005	Informatica (Abacus/Ordinamento)	Produzioni musicali
	Informatica (Mercurio)	Gestione Palestra
2004	Informatica (Ordinamento)	Portfolio dello studente
2003	Informatica (Abacus/Ordinamento)	Vivaio
	Informatica (Mercurio)	Corsi di inglese
	Informatica gestionale	Banca online
2002	Informatica (Abacus/Ordinamento)	Banca del tempo
2001	Informatica (Mercurio)	Band musicale
	Informatica gestionale	E-commerce
2000	Informatica (Abacus)	Agenzia turistica
	Informatica (Mercurio)	Concorso online
1999	Informatica (Abacus)	Agenzia immobiliare
1998	Informatica (Abacus)	Galleria d'arte

Le tracce degli anni precedenti non sono significative in quanto per le soluzioni non veniva esplicitamente richiesto l’utilizzo dei database: sono tuttavia di seguito riportate per ragioni di completezza.

Anno	Traccia
1997	Trattoria
1996	Inquinamento atmosferico
1995	Notiziario settimanale
1993	Composizione commissioni
1992	Visitatori mostra
1991	Registro di classe
1989	Impaginazione in colonne
1988	Nomi incompleti
1987	Vocabolario bilingue
1986	Analisi cliniche
1985	Editor
1984	Agenda annuale
1983	Stazione meteorologica
1982	Parole frasi
1981	Manutenzione apparato
1980	Frequenze testo
1978	Popolazione insetti
1975	Elettori voti
1974	Banconote monete

Car pooling (ITIA Ordinaria 2017)

IN QUESTA LEZIONE IMPAREREMO...

- a risolvere una traccia d'esame

Il testo

Pag. 1/3

Sessione ordinaria 2017
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

1044 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Tema di: INFORMATICA - ***Tipologia C***

Il candidato (che potrà eventualmente avvalersi delle conoscenze e competenze maturate attraverso esperienze di alternanza scuola-lavoro, stage o formazione in azienda) svolga la prima parte della prova e due tra i quesiti proposti nella seconda parte.

PRIMA PARTE

Un'azienda *start-up* vuole costruire una piattaforma Web che consenta il *car pooling* tra viaggiatori sul territorio nazionale, con l'obiettivo di diffondere l'uso di una mobilità flessibile e personalizzata in termini di percorsi e costi.
Gli utenti della piattaforma possono essere di due tipi: utenti-autisti (coloro che offrono un passaggio con la propria macchina) e utenti-passeggeri (coloro che usufruiscono del passaggio).
Gli autisti devono registrarsi sul sito ed inserire i propri dati: generalità, numero e scadenza patente di guida, dati dell'automobile utilizzata, recapito telefonico, email, fotografia.
Per ogni viaggio che intendono condividere, gli autisti devono indicare città di partenza, città di destinazione, data ed ora di partenza, contributo economico richiesto ad ogni passeggero, tempi di percorrenza stimati. È responsabilità dell'autista, mano a mano che accetterà passeggeri per un certo viaggio, dichiarare chiuse le prenotazioni per quel viaggio, utilizzando un'apposita funzione sul portale.
L'utente-passeggero si deve registrare sulla piattaforma, indicando cognome e nome, documento di identità, recapito telefonico ed email. La piattaforma fornisce ai passeggeri la possibilità di indicare città di partenza e di destinazione e data desiderata; presenta quindi un elenco di viaggi (per cui non siano ancora chiuse le prenotazioni), ciascuno con le caratteristiche dell'autista e le modalità del viaggio stesso inserite dall'autista (orario, eventuali soste previste alle stazioni di servizio, possibilità di caricare bagaglio o animali, ...).
Il passeggero sceglie quindi il viaggio desiderato con il corrispondente autista, anche esaminando il voto medio e i giudizi dei *feedback* assegnati tramite la piattaforma dai precedenti passeggeri all'autista stesso, e si prenota. Le informazioni sul passeggero vengono inviate per email dalla piattaforma all'autista scelto, il quale può consultare sul portale il voto medio e i giudizi dei *feedback* ricevuti dal passeggero da parte di precedenti autisti e decidere se accettarlo o meno. Il passeggero di conseguenza riceverà una email di accettazione o di rifiuto della prenotazione effettuata, contenente, in caso di accettazione, un promemoria con città di partenza e destinazione, data e orario del viaggio, dati dell'autista e della sua automobile.
A viaggio effettuato, il passeggero può inserire un *feedback* sull'autista, espresso sia in forma di voto numerico che di giudizio discorsivo. A sua volta, l'autista può inserire un *feedback* sul passeggero, espresso sia in forma di voto numerico che di giudizio discorsivo. Sia i voti medi che i singoli giudizi dei *feedback* ricevuti da ciascun autista sono disponibili ai passeggeri; analogamente, sia i voti medi che i singoli giudizi dei *feedback* ricevuti da ciascun passeggero sono disponibili agli autisti.

Pag. 2/3

Sessione ordinaria 2017
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

1044 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Tema di: INFORMATICA - ***Tipologia C***

Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

1. un'analisi della realtà di riferimento, giungendo alla definizione di uno schema concettuale della base di dati che, a suo motivato giudizio, sia idoneo a gestire la realtà presentata;
2. il relativo schema logico;
3. le seguenti interrogazioni espresse in linguaggio SQL:
   a) data una città di partenza, una di arrivo e una data, elencare gli autisti che propongono un viaggio corrispondente con prenotazioni non ancora chiuse, in ordine crescente di orario, riportando i dati dell'auto e il contributo economico richiesto;
   b) dato il codice di una prenotazione accettata, estrarre i dati necessari per predisporre l'email di promemoria da inviare all'utente passeggero;
   c) dato un certo viaggio, consentire all'autista di valutare le caratteristiche dei passeggeri visualizzando l'elenco di coloro che lo hanno prenotato, con il voto medio dei feedback ricevuti da ciascun passeggero, presentando solo i passeggeri che hanno voto medio superiore ad un valore indicato dall'autista;
4. il progetto di massima della struttura funzionale dell'applicazione Web, realizzando, con appropriati linguaggi a scelta sia lato client che lato server, un segmento significativo dell'applicazione che consenta l'interazione con la base di dati.

SECONDA PARTE

1. In relazione al tema proposto nella prima parte, il candidato integri il modello già realizzato al fine di gestire in automatico il numero di posti disponibili nei vari viaggi, evitando che sia responsabilità dell'autista dichiarare chiuse le prenotazioni sul portale. Nel momento in cui inserisce un viaggio, l'autista dichiara il numero massimo di posti disponibili. Mano a mano che gli autisti accettano le prenotazioni, il sistema visualizzerà solo i viaggi con posti ancora disponibili: a tal fine, una prenotazione non ancora accettata dall'autista non comporta alcun impegno del posto, che resta così ancora disponibile per prenotazioni di altri passeggeri. Per ciascun viaggio, la piattaforma mostrerà il numero dei posti disponibili e il numero delle prenotazioni non ancora accettate. Il candidato sviluppi poi la pagina web, sia lato client che lato server, per fornire ai passeggeri tali informazioni.

Pag. 3/3

Sessione ordinaria 2017
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

I044 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Tema di: INFORMATICA - ***Tipologia C***

II. In relazione al tema proposto nella prima parte, il candidato immagini di volere documentare al committente l'operatività della piattaforma proposta. A tal fine, imposti una relazione tecnica che presenti le principali caratteristiche dell'applicazione Web in termini di organizzazione e funzionalità. In particolare, imposti la struttura di tale relazione, motivando le scelte e scrivendo un esempio significativo dei relativi contenuti.

III. Dato il seguente schema relazionale:
film (id, titolo, durata, anno di produzione, genere);
attore (id, nome, cognome, data_nascita, fotografia);
recita (id_film, id_attore, ruolo);

il candidato:
- determini la modalità di gestione del campo 'fotografia' che prevede la memorizzazione di una immagine dell'attore in un formato grafico (es. JPG);
- formalizzi in linguaggio SQL lo schema fisico corrispondente allo schema relazionale, sapendo che:
  a. il campo 'genere' ammette solo i seguenti valori: fantasy, giallo, commedia, horror, drammatico, fantascienza, azione;
  b. per la relazione 'recita', i campi 'id_film' e 'id_attore' referenziano rispettivamente la chiave primaria delle relazioni 'film' e 'attore';
- discuta l'uso degli indici nel modello fisico di una base di dati e suggerisca con motivato giudizio indici appropriati per questo schema relazionale, definendoli in linguaggio SQL.

IV. Un'azienda desidera sviluppare un'applicazione Web per la prenotazione on-line di eventi culturali, fruibile sia da computer desktop che da dispositivi mobili come tablet e smartphone. Il candidato esponga i punti critici da affrontare relativamente alle differenti proprietà di visualizzazione delle varie tipologie di dispositivi e alla rispettiva fruizione dei contenuti. Illustri possibili misure risolutive, con esempi relativi all'applicazione in questione.

Durata massima della prova: 6 ore.
È consentito l'uso di manuali tecnici e di calcolatrici tascabili non programmabili.
È consentito l'uso del dizionario bilingue (italiano-lingua del paese di provenienza) per i candidati di madrelingua non italiana.
Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla dettatura del tema.

PRIMA PARTE della traccia: osservazioni

La **PRIMA PARTE** della traccia consiste nella presentazione dettagliata di una situazione che richiede come compito il progetto di un database.
Al **punto 4** viene richiesta la realizzazione di una pagina Web che effettui una interazione con database, lasciando libertà di scelta al candidato.

Analisi situazione

Il testo non richiede analisi aggiuntive, dato che la situazione è completamente descritta e ben definita e da un'attenta lettura delle specifiche si evidenzia che sono richieste le seguenti attività:
– raccolta dei dati relativi ad autisti che con il loro mezzo privato effettuano viaggi tra due città italiane;
– raccolta e pubblicazione dei dati relativi ai viaggi (data, orario ecc.);
– raccolta dei dati relativi alle prenotazione di possibili passeggeri;
– raccolta dei dati relativi ai giudizi espressi dal personale viaggiante (*feedback*);
– la traccia richiede che il sistema di gestione sia online, quindi deve essere predisposta una piattaforma adeguata a soddisfare tutte le richieste mediante un sistema client-server che permetta all'utente di interfacciarsi con il database attraverso una applicazione Web.

Ipotesi aggiuntive

Senza perdere di generalità, si ipotizza che non siano effettuate tappe intermedie: tutti i passeggeri, quindi, partono e arrivano assieme alla città destinazione del viaggio.

Schema concettuale

Il **punto 1** riguarda la progettazione dello schema concettuale della base di dati (modello E-R). Per prima cosa, realizziamo il glossario dei termini, al quale seguirà la definizione delle entità e delle relazioni che intercorrono tra di esse.

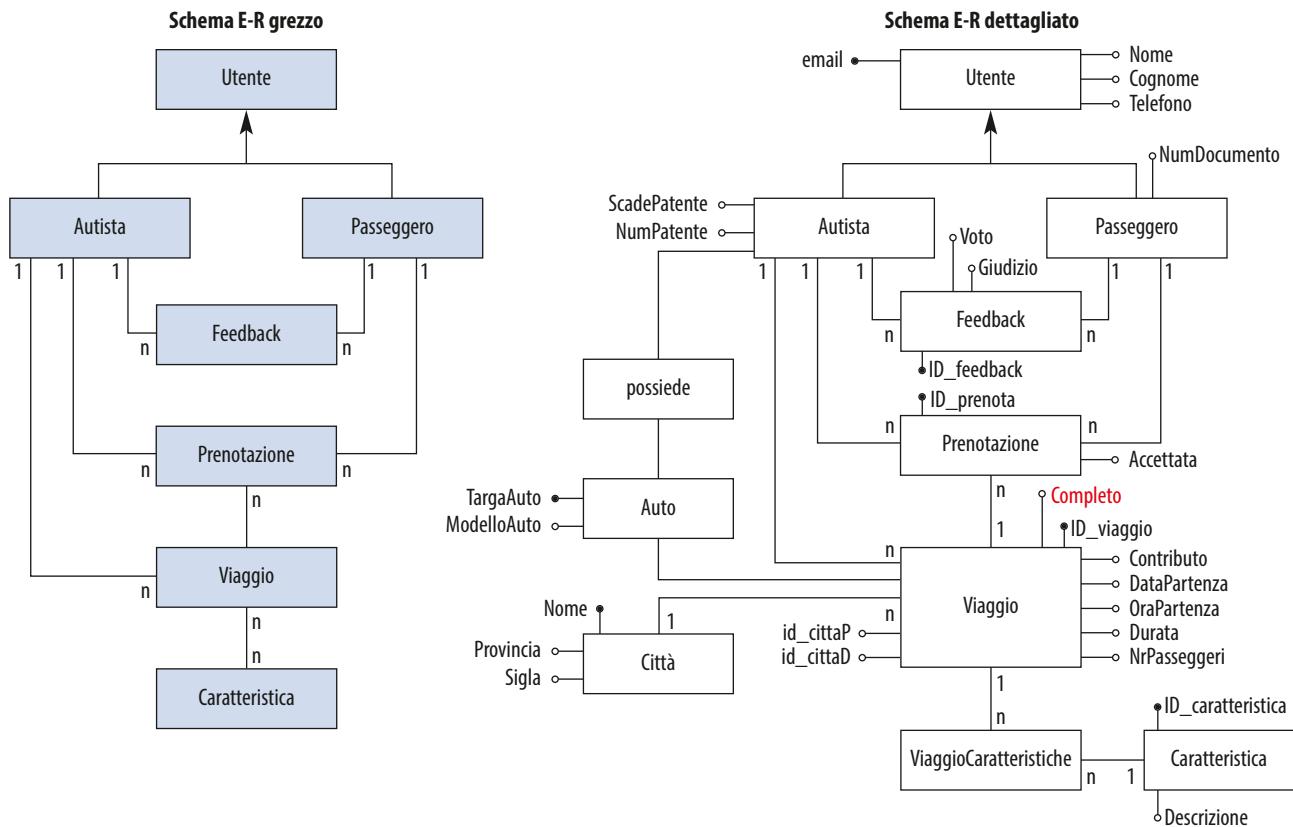
Glossario dei termini

Dalla rilettura del testo possiamo evidenziare i seguenti termini:

TERMINE	DESCRIZIONE	COLLEGAMENTI
Autista	Autista del mezzo con il quale viene effettuato il viaggio	Viaggio, Prenotazione, Feedback
Passeggero	Colui che usufruisce del servizio	Viaggio, Prenotazione, Feedback
Feedback	Giudizio dato da autisti a passeggeri e viceversa	Autista, Passeggero
Prenotazione	Impegno a partecipare a un viaggio	Viaggio, Passeggero
Viaggio	Singolo servizio effettuato da un autista tra due città	Autista, Passeggero, Città
Città	Partenza e destinazione del viaggio	Viaggio
Auto	Mezzo di trasporto con il quale si effettua il servizio	Viaggio, Optional
Caratteristica	Optional presenti e/o disponibili sull'auto	Auto, Viaggio

Partendo da questi iniziamo a definire le entità.

Entità

Dal glossario dei termini definiamo le seguenti entità.
- Utente (Autista-Passeggero): rappresenta gli utenti del servizio (sia attivi che passivi) mediante una relazione gerarchica.
- Feedback: giudizio dato da autisti a passeggeri e viceversa.
- Prenotazione: un passeggero effettua una prenotazione per un viaggio.
- Viaggio: un viaggio per una destinazione (città) ha un autista e un insieme di passeggeri.
- Città: insieme delle città italiane destinazione dei viaggi.
- Auto: rappresenta il mezzo di trasporto privato: tra gli attributi ci saranno la targa, il numero di posti e i servizi eventualmente disponibili.
- Caratteristica: optional offerti dai singoli autisti/automezzi/viaggi.

Associazioni

I legami logici esistenti tra le entità sono i seguenti:
- Utente-Feedback (1, n): un utente emette dei feedback.
- Utente-Prenotazione (1, n): un utente può effettuare più prenotazioni.
- Utente-Viaggio (1, n): un utente può effettuare più viaggi.
- Viaggio-Prenotazione (1, n): un viaggio può ottenere più prenotazioni.
- Auto-Viaggio (1, n): un auto effettua più viaggi.
- Città-Viaggio (1, n): in una città possono essere effettuati più viaggi.
- Viaggio-Caratteristiche (n, n): ogni viaggio ha più caratteristiche.
- Utente-Auto (n, n): ogni utente può utilizzare diverse auto.

Diagramma E-R

Il diagramma E-R, grezzo e dettagliato, è riportato di seguito.

La gerarchia presente è del tipo parziale ed esclusiva, in quanto un autista potrebbe anche essere passeggero e viceversa.

Unica relazione (n, n) da risolvere è quella presente tra Viaggio e Caratteristica, che deve essere affinata introducendo una entità ponte, che chiamiamo ViaggioCaratteristiche.

Per semplicità, introduciamo come chiavi primarie tutte chiavi artificiali tranne per le entità Utente, Auto, Città e ViaggioCaratteristiche, che hanno come chiave primaria rispettivamente l'indirizzo di email, la targa, il nome e la chiave composta delle due entità che collega: tutte le altre entità hanno invece una chiave primaria semplice, alla quale abbiamo dato come identificatore **ID_+ <nome_entità>**, quindi ID_viaggio, ID_città, ID_feedback ecc.

Schema logico

Il **punto 2** della traccia richiede la realizzazione dello schema logico: questo viene utilizzato per la definizione delle strutture di dati e prevede che:

– ogni entità diventi una tabella;
– un'istanza di una entità diventi una riga della tabella;
– ogni attributo dell'entità diventi una colonna della tabella;
– la chiave primaria dell'entità sia un identificatore univoco delle righe di una tabella.

Possiamo definire i seguenti schemi relazionali, indicando le chiavi primarie ed esterne per ogni entità e aggiungendo gli attributi indicati nel testo:

– Auto (Targa(pk), modello, posti)

- Utenti (email(pk), cognome, nome, telefono, telefonoVerificato, nrPatente, scadePatente, fotografia)
- Città (nome(pk), provincia, CAP)
- Viaggi (ID_viaggio(pk), dataPartenza, oraPartenza, durata, nrPosti, nrPasseggeri, completo, contributo, autista(fk) auto(fk), cittaP(fk), cittaD(fk))
- Prenotazioni (ID_prenota(pk), data, id_viaggio(fk), id_passeggero(fk), accettata)
- Feedback (ID_feedback(pk), id_utente(fk), ruolo, voto, giudizio, data)

Utilizzando come modello logico il modello relazionale sarà sufficiente implementare le tabelle relative alle singole entità: riporteremo direttamente le tabelle definite in ambiente Microsoft Access, presenti nel database carPooling2017.mdb (cartella DATABASE del CD-ROM)

Riportiamo come schema logico completo direttamente la struttura delle principali tabelle Access.

Tabella Auto

La tabella Auto ha Targa come chiave primaria.

Auto

Nome campo	Tipo dati	Descrizione
Targa	Testo	pk
modello	Testo	descrizione
posti	Numerico	

Per i nomi delle tabelle, è stata rispettata la convenzione di indicarle al plurale.
Per i nomi dei campi sono state mantenute le convenzioni:
- ID_xxx: maiuscolo come chiave primaria artificiale;
- id_xxx: minuscolo come chiave esterna.

Tabella Utenti

La tabella Utenti ha email come chiave primaria, che viene utilizzata anche al momento della registrazione per effettuare la procedura di verifica dell'identità digitale dell'utente:

Utenti

Nome campo	Tipo dati	Descrizione
email	Testo	pk
cognome	Testo	30 caratteri
nome	Testo	20 caratteri
telefono	Testo	20 caratteri
telefonoVerificato	Sì/No	per confermare identità telefonica
nrpatente	Testo	
scadePatente	Data/ora	
fotografia	Testo	link al file immagine

Tabella Citta

La tabella Citta ha nome come chiave primaria, in modo da "semplificare" le successive query di ricerca:

Citta

Nome campo	Tipo dati	Descrizione
nome	Testo	pk - nome citta
provincia	Testo	sigla provincia
CAP	Testo	

Tabella Viaggi

La tabella Viaggi ha ID_viaggio come chiave primaria artificiale autoincrement:

Viaggi

Nome campo	Tipo dati	Descrizione
ID_viaggio	Numerazione automat	pk autoincrement
data	Data/ora	
oraPartenza	Data/ora	
durata	Numerico	
nrPosti	Numerico	
nrPasseggeri	Numerico	
completo	Sì/No	
contributo	Valuta	
autista	Testo	fk- utente
auto	Testo	fk - targa auto
cittaP	Testo	fk - citta partenza
cittaD	Testo	fk - citta destinazione

Tabella Prenotazioni

La tabella Prenotazioni ha ID_prenota come chiave primaria artificiale autoincrement:

Prenotazioni

Nome campo	Tipo dati	Descrizione
ID_prenota	Numerazione automat	pk - autoincrement
data	Data/ora	data del viaggio
id_utente	Testo	fk - passeggero
id_viaggio	Numerico	fk - viaggio
confermata	Sì/No	per conferma accettazione

Tabella Feedback

La tabella Feedback ha ID_feedback come chiave primaria artificiale autoincrement:

Feedback

Nome campo	Tipo dati	Descrizione
ID_feedback	Numerazione automat	pk - autoincrement
id_utente	Testo	
ruolo	Numerico	1=passeggero 2=autista
voto	Numerico	1=pessimo 2=mediocre 3=buono 4=ottimo
commento	Testo	
data	Data/ora	

Tabelle per le caratteristiche

Per la risoluzione della relazione (n, n), introduciamo le due tabelle relative alle caratteristiche. La prima è un dizionario:

Caratteristiche

Nome campo	Tipo dati	Descrizione
ID_caratteristica	Numerazione automat	pk - autoincrement
descrizione	Testo	30 caratteri

La seconda è la tabella ponte che permette l'associazione con il viaggio:

ViaggiCaratteristiche

Nome campo	Tipo dati	Descrizione
id_viaggio	Numerico	pk viaggio
id_caratteristica	Numerico	pk caratteristica

Interrogazioni

Nel **punto 3** sono richieste tre interrogazioni di selezione.

Prima interrogazione

a) Data una città di partenza, una di arrivo e una data, elencare gli autisti che propongono

un viaggio corrispondente con prenotazioni non ancora chiuse, in ordine crescente di orario, riportando i dati dell'auto e il contributo economico richiesto.

```sql
SELECT V.data,V.[oraPartenza], V.autista, V.[contributo], A.modello, A.targa
FROM Viaggio AS V, Auto AS A
WHERE V.auto = A.targa AND V.cittaP=[citta_partenza] AND V.cittaD=[citta_destinazione]
  AND  V.completo=0 AND V.data='data_voluta'
    ORDER BY V.oraPartenza ASC
```

Inserendo nel nostro database per esempio "Milano" come [citta_partenza], "Genova" come [citta_destinazione] e il giorno "04-07-2017" come [data_voluta] otteniamo il seguente risultato:

1 Viaggio Query

data	oraPartenza	autista	contributo	modello	targa
04/07/2017	18:00:00	rossi@xxx.it	€ 5,00	Suv	QW111QW
04/07/2017	18:30:00	gialli@xxx.it	€ 5,00	Berlina	AA111SS

b) Dato il codice di una prenotazione accettata, estrarre i dati necessari per predisporre l'email di promemoria da inviare all'utente passeggero.

```sql
SELECT P.id_utente, V.data, V.cittaP, v.cittaD, V.[oraPartenza],
       V.autista, V.[contributo], A.modello, A.targa
FROM Viaggi AS V, Prenotazioni AS P, Auto AS A
WHERE V.ID_viaggio = P.id_viaggio AND V.auto = A.targa
   AND P.ID_prenota=[inserire ID prenotazione] AND P.confermata<>0;
```

Inserendo nel nostro esempio "6" come ID_prenotazione otteniamo il seguente risultato:

2 Mail Conferma Query

id_utente	data	cittaP	cittaD	oraParten	autista	contributo	modello	targa
verdi2@xxx.it	22/06/2017	Milano	Genova	11:00:00	rossi@xxx.it	€ 3,00	Suv	QW111QW

c) Dato un certo viaggio, consentire all'autista di valutare le caratteristiche dei passeggeri visualizzando l'elenco di coloro che lo hanno prenotato, con il voto medio dei feedback ricevuti da ciascun passeggero, presentando solo i passeggeri che hanno voto medio superiore a un valore indicato dall'autista.

```sql
SELECT F.id_utente AS Passeggero, U.cognome, U.nome,  AVG(F.voto) AS 'media'
FROM  Feedback AS F, prenotazione AS P,viaggio AS V, Utenti AS U
WHERE V.id_viaggio=[inserire codice viaggio] AND V.id_viaggio=P.id_viaggio
  AND P.id_utente=F.id_utente  AND U.email=F.id_utente AND F.ruolo=2
    GROUP BY F.id_utente, U.cognome, U.nome
    HAVING AVG(F.voto) > [media minima]
```

Inserendo nel nostro database per esempio "1" come ID_viaggio e "2" come media desiderata, otteniamo il seguente risultato:

3 Feedback Query

Passeggero	cognome	nome	media
rosa@ccc.it	rosa	ciro	3
verdi2@xxx.it	verdi	rosa	4

Conclusioni e migliorie

La soluzione proposta non è sicuramente la migliore né la più efficiente, ma è stata adottata tenendo conto delle ore a disposizione per la prova scritta e delle richieste specifiche della traccia.

Una semplice (e doverosa) miglioria riguarda la struttura della tabella Città, che "non è normalizzata" e quindi richiede di essere messa in 3 FN introducendo la nuova entità Province.

Anche la tabella Feedback richiede l'introduzione di due dizionari per classificare i Ruoli che hanno gli utenti (1 = passeggero 2 = autista) e i Giudizi (stabiliti in: 1 = pessimo 2 = mediocre 3 = buono 4 = ottimo).

Il diagramma E-R completo è quindi il seguente:

Il database così completato è memorizzato nel file carPooling2017.accdb presente nella cartella DATABASE del CD-ROM allegato al volume; in alternativa, può essere scaricato dalla cartella MATERIALI presente sia nel CD-ROM che nella nella sezione del sito www.hoepliscuola.it riservata a questo volume.

Segmento di codice Web

Al **punto 4** non viene richiesto una segmento esplicito di codice lato server, ma la scelta viene lasciata al candidato: è sufficiente che il segmento scelto sia «significativo e che consenta l'interazione con la base di dati».

La nostra scelta ricade sulla parte di codice indicata tra le funzionalità richieste nel testo e cioè: «la piattaforma fornisce ai passeggeri la possibilità di indicare città di partenza e di destinazione e data desiderata; presenta quindi un elenco di viaggi (per cui non siano ancora chiuse le prenotazioni), ciascuno con le caratteristiche dell'autista e le modalità del viaggio stesso inserite dall'autista (orario, eventuali soste previste alle stazioni di servizio, possibilità di caricare bagaglio o animali, ...)».

Realizziamo quindi in un'unica pagina php, la pagina mostra in due ComboBox l'elenco delle città per la scelta della città di partenza e di destinazione e, tramite la tecnica del postback, l'elenco dei viaggi disponibili presenti nella tabella Viaggi del database.

Alla prima esecuzione della pagina1.php viene caricato dalla tabella Citta del database l'elenco delle possibili alternative, sia come città di partenza che di destinazione del viaggio.

Connessione ad Access

Nella prima parte dello script il codice HTML visualizza la testata della pagina con il logo della agenzia, quindi il codice php e infine effettua la connessione al database carPooling2017.mdb includendo il file connettiCP.php riportato a pagina seguente.

```
<?php
  // connette al database Access  dsn-less con PHP
  $percorso = "../database/";          // relativo da dove è la pagina php
  $database = "carPooling2017.mdb";

  // percorso del database
  $archivio = realpath($percorso.$database);

  // Stringa di connessione ADO DB
  $sc = "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=".$archivio.";";

  // Creo due oggetti COM contenenti gli oggetti Connection e Recordset
  $cn = new COM("ADODB.Connection") or die("Non va ADO");
  $rs = new COM("ADODB.Recordset");
?>
```

Il linguaggio php permette di utilizzare le librerie messe a disposizione da Windows tramite la direttiva new COM(): noi ci connetteremo quindi a un database MS Access tramite la libreria ADODB. Dalla root del nostro sito creiamo una cartella database e definiamo tre variabili che ci permettono di individuare il nostro file:

– $percorso;
– $database;
– $archivio.

In seguito basterà sostituire il nome nella variabile $database mantenendo invariato il resto dello script e richiamandolo con la clausola:

```
<?php
// effettuo la connessione ADO ad access
include 'connettiCP.php';
```

Se lo script va a buon fine, viene effettuata la connessione e vengono messi a disposizione del programmatore i due oggetti $cn e $rs per poter effettuare le query e recuperare i dati “di ritorno” dal server.

Forma di connessione ai database con MySQL

```
<?php
// connette al SQL server sulla macchina locale con credenziali:
// utente 'prova', password '12345'
$host     = 'localhost';      // ipotizzando di accedere ad un server locale.
$database = 'carPooling2017';
$utente   = 'prova';
$password = '12345';


$con = new mysqli($host, $utente , $password, $database);

// prova la connessione al database
 if (mysqli_connect_errno()) {
    echo("Connessione non effettuata: ".mysqli_connect_error()."<BR>");
    exit();
 }

?>
```

Naturalmente, affinché la connessione abbia successo, è necessario che l'amministratore del database garantisca l'accesso all'utente "prova", che può essere fatta manualmente, oppure con la seguente istruzione DCL:

```
GRANT SELECT,INSERT,UPDATE,DELETE
ON 'carPooling2017'.*
TO 'prova@localhost'
IDENTIFIED BY '12345'
```

Ora è possibile utilizzare lo script di connessione, inserendo il seguente segmento di codice:

```
<?php
// termina se l'inclusione fallisce
require 'connettiCP.php'
…

?>
```

La trattazione riporta il codice che utilizza il database Access.

Selezione città e scelta viaggi

La prima parte dello script consiste in un insieme di tag HTML di apertura con la visualizzazione di due immagini "di cortesia" e la predisposizione della tabella per contenere i ComboBox:

```
<HTML xmlns="http://www.w3.org/1999/xhtml">
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</HEAD>
<BODY>
<p>
<b><img src="../images/logoeditore.gif" width="364" height="56">  </b></p>
<p><img src="../images/carPooling.jpg" width="538" height="246"> </p>
<p><strong> </strong></p>
<table width="400" border="1">
  <tr>
    <th width="192" scope="col"><strong>Citta di partenza</strong></th>
    <th width="198" scope="col"><strong>Citta di destinazione </strong></th>
  </tr>
```

Dopo aver aperto la connessione, si effettua una semplice query di selezione dalla tabella Citta:

```
<?php
// effettuo la connessione ADO ad access
include 'connettiCP.php';
// tecnica POSTBACK per realizzare il form nella stessa pagina
if (!$_POST)  // prima volta della sua esecuzione
{
 $query1 = " SELECT nome FROM citta  ";
  // apro la Connection
  $cn->Open($sc);
  // eseguo la query generando il Recordset
  $rs->Open($query1, $cn);
  // primo combo box
  // prima riga tabella
  echo "  <tr>";
  echo "    <th scope='col'><select name='partenza' onchange='aggiornaParti(this)'>";
  // ciclo di lettura e stampa del recordset
  while (!$rs->eof)
  {                     // inserisce un dato nel combo box
    echo "<option value=".$rs->fields[0]->value.">".$rs->fields[0]->value."</option>";
    $rs->movenext();    // metodo movenext() per passare al record successivo
  }
  echo "    </select></th>";
  $rs->Close();
```

Selezionando le città, le scelte vengono riportate in due caselle di testo e, di seguito, l'utente conferma la scelta mediante il pulsante che richiama la stessa pagina:

```
// stampa del form
$form="<FORM name='form1' ACTION='".$_SERVER['PHP_SELF']."' METHOD=POST><TABLE><TR>
<INPUT TYPE='submit' VALUE=' Cerca viaggio'>
<input name='citta1' type='text' readonly='true' >
<input name='citta2' type='text' readonly='true' >
</TABLE>";
echo $form;
```

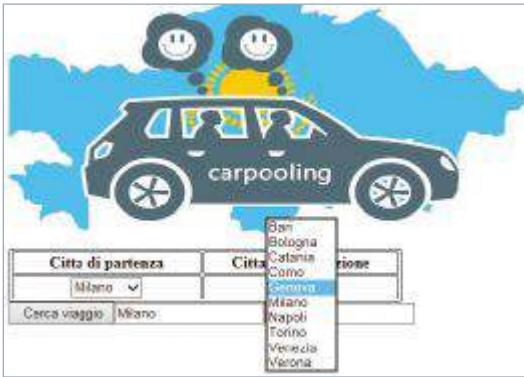
Viene utilizzata una funzione javascript per riportare la scelta selezionata dal ComboBox in un TextBox:

```
<SCRIPT type="text/javascript">
 function aggiornaParti(sel){
  var f = document.form1;
  f.citta1.value = sel.options[sel.selectedIndex].value;
  f.citta1.value = sel.options[sel.selectedIndex].text;
 }
```

Viene quindi letto il parametro scelta dalla stringa di query, verificando se questa variabile è stata settata, oppure se la pagina presente è stata caricata direttamente digitandone l’indirizzo nel browser:

```
// apro la Connection
$cn->Open($sc);
// predispongo l'interrogazione
$query2="
  SELECT ID_Viaggio AS campo1, data AS campo2, oraPartenza AS campo3, durata AS campo4, autista AS campo5,
    auto AS campo6
  FROM viaggi
  WHERE cittaP='".$citta1."' AND cittaD ='".$citta2."'AND completo<>true
 ";
// eseguo le query
$rs->Open($query2, $cn);
// stampa tabella risultati con riga di intestazione
echo "<TABLE><TR><TH>ID_Viaggio<TH>Data <TH>ora <TH>durata <TH>autista<TH>targa auto";
```

Prepariamo ora la stringa che contiene l’interrogazione che dovrà essere effettuata in SQL: a titolo di esempio, viene assegnato un alias ad alcuni dati, che sarà l’identificatore col quale potranno essere individuati nel recordSet di risposta del server:

```
if ($rs->eof) // trovato accetta credenziali
   echo '<TR><br>Nessun viaggio disponibile per questo itinerario';
else
  // ciclo di letture e stampa del recordset
  while (!$rs->eof)
  {
    echo "<TR>";
    // stampa del singolo campo in ogni cella
    echo'<TD>'.$rs->fields['campo1']->value."</TD>";
    echo'<TD>'.$rs->fields['campo2']->value."</TD>";
    echo'<TD>'.$rs->fields['campo3']->value."</TD>";
    echo'<TD>'.$rs->fields['campo4']->value."</TD>";
    echo'<TD>'.$rs->fields['campo5']->value."</TD>";
    echo'<TD>'.$rs->fields['campo6']->value."</TD>";
    echo'</TR>';
    // metodo movenext() per passare al record successivo
    $rs->movenext();
  }
echo "</TABLE>";
// chiusura connessione
$cn->close();
```

I campi nel recordSet possono essere in ogni caso individuati in base alla loro posizione fisica nell’array, come per esempio l’ID_viaggio, che occupa la $rs –> fields[0], oppure mediante il loro identificatore definito nella query, cioè per esempio $rs –> fields [‘campo1’].

Le due videate del nostro lavoro sono riportate a pagina seguente: nella prima l’utente seleziona le città di partenza e di destinazione, che vengono confermate mediante il pulsante Cerca viaggio: come risposta dell’interrogazione viene proposta la seconda videata, che contiene i risultati della query.

Citta di partenza	Citta di destinazione
Milano	Genova

ID_Viaggio	Data	ora	durata	autista	targa auto
2	22/06/2017	11:00:00	120	rossi@xxx.it	QW111QW
6	21/07/2017	09:00:00	70	gialli@xxx.it	AA111SS
8	02/07/2017	12:00:00	80	rossi@xxx.it	QW111QW
9	29/07/2017	15:00:00	75	gialli@xxx.it	AA111SS
10	04/07/2017	18:00:00	80	rossi@xxx.it	QW111QW
11	04/07/2017	18:30:00	80	gialli@xxx.it	AA111SS
12	04/07/2017	18:30:00	80	gugi@xxx.it	BB22233

Il “minisito“ completo è raggiungibile all’indirizzo http://localhost/maturita/2017_CP.

SECONDA PARTE della traccia

La **SECONDA PARTE** della traccia può essere divisa in tre parti:
– i primi due quesiti sono “il seguito” del progetto richiesto nella **PRIMA PARTE**;
– il terzo quesito consiste in un nuovo “mezzo esercizio”, completato da una domanda teorica;
– il quarto quesito è una domanda teorica che richiede spiegazioni sulle tecniche responsive.

Integrazione della PRIMA PARTE con automatismi

La **prima domanda** della **SECONDA PARTE** della traccia richiede al candidato di integrare quanto realizzato nella **PRIMA PARTE** introducendo degli automatismi software realizzabili con opportune query del tipo UPDATE/SET/WHERE, che vengono richiamate al verificarsi di particolari situazioni, come per esempio quando il numero di posti occupati risulta essere uguale al numero di posti liberi.
Come esempio, riportiamo la soluzione adottata per indicare che un viaggio è “chiuso”: abbiamo aggiunto un attributo nrPasseggeri che viene incrementato ogni volta che una prenotazione viene confermata e confrontato con il valore presente nell’attributo nrPosti: se tali valori sono uguali, viene settato il flag completo al valore true.

La documentazione del progetto della PRIMA PARTE

La **seconda domanda** della **SECONDA PARTE** della traccia richiede al candidato di documentare al cliente il progetto realizzato nella prima parte mediante una relazione tecnica contenente le principali caratteristiche dell’applicazione: possiamo organizzare la relazione nei seguenti paragrafi, che rispecchiano le fasi della progettazione a cascata:
a) studio di fattibilità e raccolta dei requisiti;
b) analisi;
c) progettazione;
d) sviluppo e realizzazione;
e) verifica e collaudo;
f) manutenzione.

Per ciascuna di esse indicheremo le scelte effettuate riportando schemi e diagrammi e i limiti e gli estremi di ogni dominio applicativo.

Mini progetto film/attore/recita

La **terza domanda** si articola in tre richieste relative allo schema relazionale indicato: per quanto riguarda la **prima richiesta**, anche se è possibile immagazzinare immagini all'interno di quasi tutti i database presenti in commercio, tale soluzione è sconsigliata e si preferisce introdurre nelle tabelle un campo dove viene indicato il collegamento al file immagine (riportando il percorso e il nome del file), che viene caricato in una apposita directory Immagini predisposta per tale tipologia di file.

La **seconda richiesta** consiste nella descrizione SQL dello schema fisico con alcuni vincoli indicati; la soluzione è riportata di seguito:

```
CREATE TABLE film
(
  ID INT PRIMARY KEY,
  titolo VARCHAR(100),
  annoProduzione  VARCHAR(4),
  genere VARCHAR(20) CHECK (genere IN ('fantasy','giallo', ..., 'azione'))
);
```

Alcuni database non ammettono la clausola CHECK: in tale situazione, un workaround si ottiene inserendo una tabella Generi, in cui vengono inserite tutte le alternative previste come unico attributo (che fa anche da chiave primaria) e imponendo sull'attributo genere della tabella Film un CONSTRAINT come quello indicato di seguito:

```
CREATE TABLE film
(
  ID INT PRIMARY KEY,
  titolo VARCHAR(100),
  annoProduzione VARCHAR(4),
  genere VARCHAR(20) NOT NULL CONSTRAINT a REFERENCES Generi(descrizione)
);
```

È inoltre richiesta la scrittura di una query per la creazione della tabella Recita:

```
CREATE TABLE Recite
(
  id_film INT CONSTRAINT x REFERENCES film(ID),
  id_attore INT CONSTRAINT y REFERENCES attore(ID),
  ruolo VARCHAR(20),
  PRIMARY KEY (id_film,id_attore)
);
```

La **terza richiesta** consiste nella descrizione dell'utilità e delle funzionalità dei file indice: un indice è una sorta di schedario, quindi una tabella essa stessa, che tiene traccia di dove sono posizionati i dati all'interno del database, in modo da poter effettuare su di essi un accesso diretto.
I file indice possono quindi essere utilizzati nel momento in cui vengono eseguite delle query per accelerare la loro esecuzione, in quanto le operazioni di ricerca vengono effettuate consultando l'indice per identificare la posizione esatta occupata dalle informazioni sul database stesso.

Una seconda funzione svolta dall'indice è quella di ordinare logicamente la tabella: creare, per esempio, un indice alfabetico come quello realizzato attraverso la seguente istruzione SQL permette di visualizzare l'elenco dei Film in ordine di titolo:

```
CREATE INDEX indiceFilm ON Film(Titolo)
```

Nel seguente esempio, invece, viene creato un indice di più attributi sulla tabella Attori, in modo da realizzare un ordinamento composto tra cognome e nome:

```
CREATE INDEX indiceAttori ON Attori(cognome, nome)
```

Problematiche nei siti Web per dispositivi mobili

La **quarta domanda** della **SECONDA PARTE** della traccia richiede al candidato di descrivere come le tecnologie informatiche permettano di usufruire dei contenuti digitali mediante apparecchiature diverse e dispositivi mobili: si devono descrivere i punti critici da affrontare relativamente alle differenti proprietà di visualizzazione delle varie tipologie di dispositivi e alla rispettiva fruizione dei contenuti, nonché le tecniche che si utilizzano per la realizzazione di pagine Web con design responsivo, o Responsive Web Design (RWD), in grado di adattarsi graficamente in modo automatico al dispositivo sul quale vengono visualizzati (computer con diverse risoluzioni, tablet, smartphone, cellulari, Web TV) e riducendo al minimo la necessità dell'utente di ridimensionare e scorrere i contenuti.

Il ridimensionamento è sempre problematico in presenza di immagini, in quanto la loro qualità è legata al numero di pixel e, quindi, alla loro dimensione: si può risolvere a monte questo problema utilizzando solo immagini vettoriali.

Eventi dal vivo (ITIA Ordinaria 2015)

AREA DIGITALE

LEZIONE 4
Trasporto passeggeri (City2City) (ITSI Ordinaria 2016)

LEZIONE 5
Abbigliamento (Gamma) (ITSI Esempio 2016)

IN QUESTA LEZIONE IMPAREREMO...

- a risolvere una traccia d'esame

Il testo

Pag. 1/3

Sessione ordinaria 2015
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

M963 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Tema di: INFORMATICA

Il candidato svolga la prima parte della prova e risponda a due tra i quesiti proposti nella seconda parte.

PRIMA PARTE

Si vuole realizzare una web community per condividere dati e commenti relativi a eventi dal vivo di diverse categorie, ad esempio concerti, spettacoli teatrali, balletti, ecc. che si svolgono in Italia.

Gli eventi vengono inseriti nel sistema direttamente dai membri stessi della community, che si registrano sul sito fornendo un nickname, nome, cognome, indirizzo di e-mail e scegliendo una o più categorie di eventi a cui sono interessati.

Ogni membro iscritto riceve periodicamente per posta elettronica una newsletter, emessa automaticamente dal sistema, che riporta gli eventi delle categorie da lui scelte, che si svolgeranno nella settimana seguente nel territorio provinciale dell'utente.

I membri registrati possono interagire con la community sia inserendo i dati di un nuovo evento, per il quale occorre specificare categoria, luogo di svolgimento, data, titolo dell'evento e artisti coinvolti, sia scrivendo un post con un commento ed un voto (da 1 a 5) su un evento.

Il sito della community offre a tutti, sia membri registrati sia utenti anonimi, la consultazione dei dati on line, tra cui:

- visualizzazione degli eventi di un certo tipo in ordine cronologico, con possibilità di filtro per territorio di una specifica provincia
- visualizzazione di tutti i commenti e voti relativi ad un evento.

Il candidato, fatte le opportune ipotesi aggiuntive, sviluppi:

1. un'analisi della realtà di riferimento individuando le possibili soluzioni e scelga quella che a suo motivato giudizio è la più idonea a rispondere alle specifiche indicate
2. uno schema concettuale della base di dati
3. uno schema logico della base di dati
4. la definizione in linguaggio SQL di un sottoinsieme delle relazioni della base di dati in cui siano presenti alcune di quelle che contengono vincoli di integrità referenziale e/o vincoli di dominio, laddove presenti

Pag. 2/3

Sessione ordinaria 2015
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

M963 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Tema di: INFORMATICA

5. le seguenti interrogazioni espresse in linguaggio SQL:
   a. elenco degli eventi già svolti, in ordine alfabetico di provincia
   b. elenco dei membri che non hanno mai inserito un commento
   c. per ogni evento il voto medio ottenuto in ordine di categoria e titolo
   d. i dati dell'utente che ha registrato il maggior numero di eventi
6. il progetto della pagina dell'interfaccia WEB che permetta ad un utente registrato di svolgere le operazioni specificate
7. la codifica in un linguaggio a scelta di un segmento significativo dell'applicazione Web che consente l'interazione con la base di dati.

SECONDA PARTE

Il candidato (che potrà eventualmente avvalersi delle conoscenze e competenze maturate attraverso esperienze di alternanza scuola-lavoro, stage o formazione in azienda) risponda a due quesiti a scelta tra quelli sotto riportati:

I. In relazione al tema proposto nella prima parte, descriva in che modo è possibile integrare lo schema concettuale sopra sviluppato in modo da poter gestire anche inserzioni pubblicitarie. Ogni inserzione è costituita da un testo e un link e può essere correlata a una o più categorie di eventi in modo da essere visualizzata in funzione dei contenuti visitati e delle preferenze degli utenti.

II. In relazione al tema proposto nella prima parte, progetti un layout di pagina idoneo a garantire un aspetto grafico comune a tutte le pagine dell'applicazione e ne codifichi alcuni elementi in un linguaggio per la formattazione di pagine Web a sua scelta.

Pag. 3/3

Sessione ordinaria 2015
Seconda prova scritta

Ministero dell'Istruzione, dell'Università e della Ricerca

M963 – ESAME DI STATO DI ISTRUZIONE SECONDARIA SUPERIORE

Indirizzo: ITIA - INFORMATICA E TELECOMUNICAZIONI
ARTICOLAZIONE INFORMATICA

Tema di: INFORMATICA

III. Si consideri la seguente tabella:

Cognome	Nome	Telefono	Livello	Tutor	Tel-tutor	Anticipo versato
Verdi	Luisa	345698741	avanzato	Bianca	334563215	100
Neri	Enrico	349523698	avanzato	Carlo	369852147	150
Rossi	Rosa	347532159	base	Alessia	333214569	120
Bianchi	Paolo	341236547	base	Carlo	369852147	150
Rossi	Mario	349567898	base	Carlo	369852147	90
Neri	Enrico	349523698	complementi	Dina	373564987	100

Il candidato verifichi le proprietà di normalizzazione e proponga uno schema equivalente che rispetti la 3ª Forma Normale, motivando le scelte effettuate.

IV. Nella formalizzazione di uno schema concettuale, le associazioni tra entità sono caratterizzate da una cardinalità: esponga il significato e la casistica che si può presentare.

Durata massima della prova: 6 ore.
È consentito l'uso di manuali tecnici (references riportanti solo la sintassi, non guide dei linguaggi utilizzati).
È consentito l'uso del dizionario bilingue (italiano-lingua del paese di provenienza) per i candidati di madrelingua non italiana.
Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla dettatura del tema.

PRIMA PARTE della traccia: osservazioni

La **PRIMA PARTE** della traccia consiste nella presentazione dettagliata di una situazione che richiede come compito il progetto di un database affinché una Web community possa condividere eventi che si svolgono sul territorio nazionale.
Rispetto alle altre tracce, le richieste sono più dettagliate e articolate in 7 punti, dove il settimo punto richiede lo sviluppo di un "qualunque" segmento di codice che interagisca con il database, quindi lasciando libera scelta al candidato.

Analisi situazione

Il **punto 1** richiede l'analisi della situazione con la proposta di soluzioni alternative.
Il testo, sufficientemente completo, chiede di sviluppare le seguenti funzionalità:
– registrazione dell'utente alla Web community, con indicazione delle categorie degli eventi ai quali è interessato a partecipare;
– inserimento degli eventi da parte del membro promotore;
– inserimento dei feedback sugli eventi da parte di un membro per esprimere una valutazione;
– visualizzazione degli eventi che si terranno in una certa provincia;
– visualizzazione dei commenti e delle valutazioni di un certo evento;
– comunicazione periodica a tutti i membri degli eventi che possano interessarli.

Essendo una Web community, l'indicazione della soluzione è "palese" nel testo, cioè si deve realizzare un database che possa essere condiviso nel Web da tutti i membri che si registrano allo stesso, sempre mediante accesso Web.
Le alternative per l'interazione tra membro e database sono sostanzialmente due:
– da postazione fissa;
– da dispositivo mobile.

Ne risulta che l'applicazione sarà articolata in due componenti, la prima che richiederà la realizzazione di un sito Web collocato su un server nel quale verrà inserito il database, e una seconda che sarà una app che ogni utente scaricherà sul proprio dispositivo mobile per poter effettuare le operazioni sopra elencate.

Nell'applicazione lato server deve essere prevista la presenza di un moderatore che effettui il controllo dei dati inseriti ed eventualmente rimuova i commenti sconvenienti: inoltre, sempre lato server, deve essere registrato un trigger che, a scadenza mensile, invii automaticamente le mail contenenti la newsletter personalizzata con gli eventi programmati.

Ipotesi aggiuntive

Anche se non esplicitamente dichiarato, si ipotizza che, contestualmente alla registrazione, gli utenti forniscano una password per l'accesso; si suppone che le categorie degli eventi siano fissate dal moderatore.

Schema concettuale

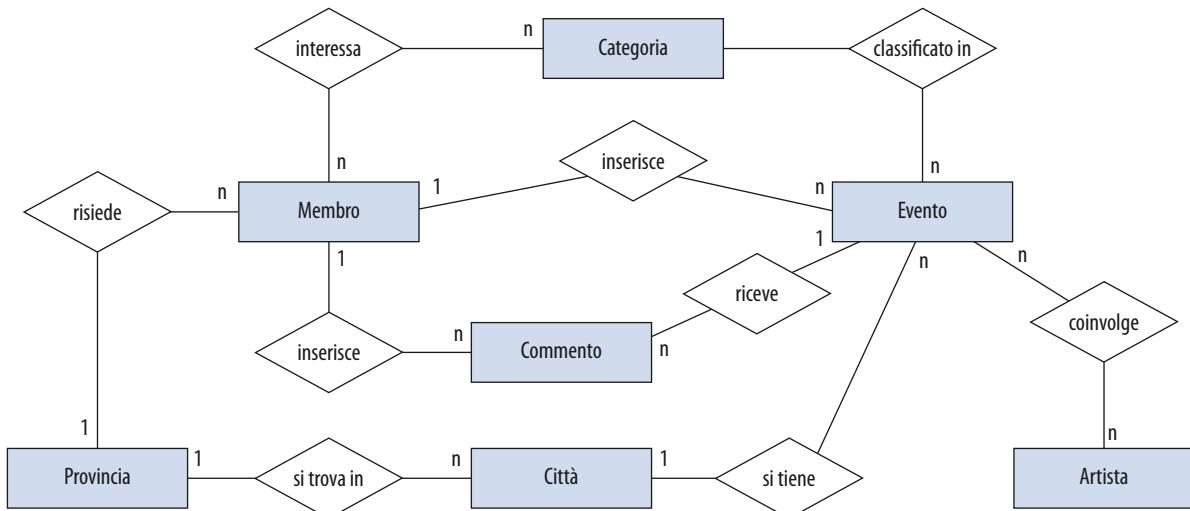
Il **punto 2** riguarda la progettazione dello schema concettuale della base di dati (modello E-R).
Per prima cosa, realizziamo il glossario dei termini, al quale seguirà la definizione delle entità e delle relazioni che intercorrono tra di esse.

Glossario dei termini

Dalla rilettura del testo possiamo evidenziare i seguenti termini:

Termine	Descrizione	Collegamenti
Evento	Spettacolo dal vivo che si svolge in italia	Categoria, Membro, Città
Categoria	Classifica l'evento (concerti, spettacoli teatrali, balletti ecc.)	Evento
Membro	Iscritto alla community che interagisce inserendo eventi e commenti	Evento, Commento, Provincia
Città	Luogo dove si tiene l'evento	Evento, Provincia
Provincia	Zona di territorio che raggruppa più città	Città
Artista	Attore, musicista, ecc. protagonista dell'evento	Evento
Commento	Giudizio con voto (1-5) e descrizione espresso da un membro	Evento, Membro

Entità

Dal glossario dei termini definiamo le seguenti entità.
- Evento: rappresentazione.
- Categoria: tipologie dell'evento.
- Città: luogo dove si tiene l'evento.
- Provincia: provincia del territorio nazionale.
- Artista: entità che rappresenta un protagonista dell'evento.
- Membro: rappresenta gli utenti della Web community.
- Commento: giudizio dato da un membro a un evento.

Associazioni

I legami logici esistenti tra le entità sono i seguenti.
- Categoria-Evento (1, n): più eventi appartengono alla stessa categoria.
- Evento-Commento (1, n): per ogni evento sono presenti più commenti.
- Commento-Membro (1, n): per ogni evento un membro può esprimere un commento.
- Artista-Evento (n, n): un artista può partecipare a più eventi e a ogni evento possono esserci più artisti.
- Membro-Categoria (n, n): ogni membro può essere interessato a più categorie di eventi e ogni categoria di eventi può interessare più membri.
- Città-Evento (1, n): in una città possono esserci più eventi.
- Membro-Evento (1, n): un membro può inserire più eventi.
- Provincia-Città (1, n): in una provincia ci possono essere più città.

Diagramma E-R

Il diagramma E-R grezzo è quindi il seguente:

Non ci sono gerarchie da risolvere, ma sono presenti due relazioni (n, n) tra:
– Membro e Categoria, che viene affinata introducendo una entità ponte chiamata MembriCategorie:
  • Membro-MembriCategorie (1, n)
  • Categoria-MembriCategorie (1, n)
– Evento e Artista, che viene affinata introducendo una entità ponte chiamata ArtistiEventi:
  • Artista-ArtistiEventi (1, n)
  • Evento-ArtistiEventi (1, n)

Per semplicità, introduciamo come chiavi primarie tutte chiavi artificiali: tranne le entità ponte che hanno come chiave primaria la chiave composta delle due entità che collegano, tutte le altre entità hanno una chiave primaria semplice, alla quale abbiamo dato come identificatore **ID_+<nome_entità>**, quindi ID_artista, ID_evento, ID_membro ecc.

Schema logico

Il **punto 3** richiede di derivare dallo schema concettuale lo schema logico, che viene utilizzato per la definizione delle strutture di dati e prevede che:
– ogni entità diventi una tabella;
– un'istanza di una entità diventi una riga della tabella;
– ogni attributo dell'entità diventi una colonna della tabella;
– la chiave primaria dell'entità sia un identificatore univoco delle righe di una tabella.

Possiamo definire i seguenti schemi relazionali, indicando le chiavi primarie ed esterne per ogni entità e aggiungendo gli attributi indicati nel testo:
– Eventi (ID_evento(pk), descrizione, data , id_città(fk), id_membro(fk),id_categoria(fk))
– Categorie (ID_categoria(pk), descrizione)
– Citta (ID_citta(pk), citta, id_provincia(fk))
– Province (ID_provincia(pk), provincia, id_regione(fk))
– Regioni (ID_regione(pk), regione)
– Artisti (ID_artista(pk), cognome, nome)
– Membri (ID_membro(pk), cognome, nome, nickname, email, password, id_provincia(fk))
– Commenti (ID_commento(pk), commento, voto, data , id_evento(fk), id_membro(fk))

Utilizzando come modello logico il modello relazionale sarà sufficiente implementare le tabelle relative alle singole entità: riporteremo direttamente le tabelle definite in ambiente Microsoft Access, presenti nel database Eventi2015.mdb.

Riportiamo come schema logico completo direttamente la struttura delle principali tabelle Access.

Tabella Eventi

La tabella Eventi ha ID_evento come chiave primaria artificiale autoincrement:

Eventi

Nome campo	Tipo dati	Descrizione
ID_evento	Numerazione automat	pk
id_citta	Numerico	fk citta
id_membro	Numerico	fk membro che lo inserisce
id_categoria	Numerico	fk categorie
descrizione	Testo	80 caratteri
data	Data/ora	formato gg:mm:aaaa

Per i nomi delle tabelle è stata rispettata la convenzione di indicarla al plurale.
Per i nomi dei campi sono state mantenute le convenzioni:
– ID_xxx : maiuscolo come chiave primaria artificiale;
– id_xxx : minuscolo come chiave esterna.

Tabella Categorie

La tabella Categorie ha ID_categoria come chiave primaria artificiale autoincrement:

Categorie

Nome campo	Tipo dati	Descrizione
ID_categoria	Numerazione automat	
descrizione	Testo	40 caratteri

Tabella Citta

La tabella Citta ha ID_citta come chiave primaria artificiale autoincrement:

Citta

Nome campo	Tipo dati	Descrizione
ID_citta	Numerico	
citta	Testo	20 caratteri
id_provincia	Numerico	fk province

Tabella Province

La tabella Province ha ID_provincia come chiave primaria artificiale autoincrement:

Province

Nome campo	Tipo dati	Descrizione
ID_provincia	Numerico	
provincia	Testo	
id_regione	Numerico	

Si è introdotta "in itinere" una tabella aggiuntiva Regioni, così da completare il database: nella tabella Province è quindi presente la chiave esterna id_regione, che collega ogni provincia alla regione di appartenenza.

Tabella Artisti

La tabella Artisti ha ID_artista come chiave primaria artificiale autoincrement:

Artisti

Nome campo	Tipo dati	Descrizione
ID_artista	Numerazione automat	
cognome	Testo	30 caratteri
nome	Testo	20 caratteri

Anche la tabella artisti potrebbe essere completata con attributi che permettano di dettagliare e classificare l'artista ma, dato che non è richiesto esplicitamente dal testo, ci limitiamo a inserire solo cognome e nome, lasciando il completamento dell'entità con gli ulteriori attributi come esercizio per lo studente.

Tabella Membri

La tabella Membri ha ID_membro come chiave primaria artificiale autoincrement:

Membri

Nome campo	Tipo dati	Descrizione
ID_membro	Numerico	pk
nome	Testo	30 caratteri
cognome	Testo	40 caratteri
codFiscale	Testo	16 caratteri
nickname	Testo	20 caratteri
password	Testo	12 caratteri
email	Testo	30 caratteri
id_provincia	Testo	fk provincia dove risiede

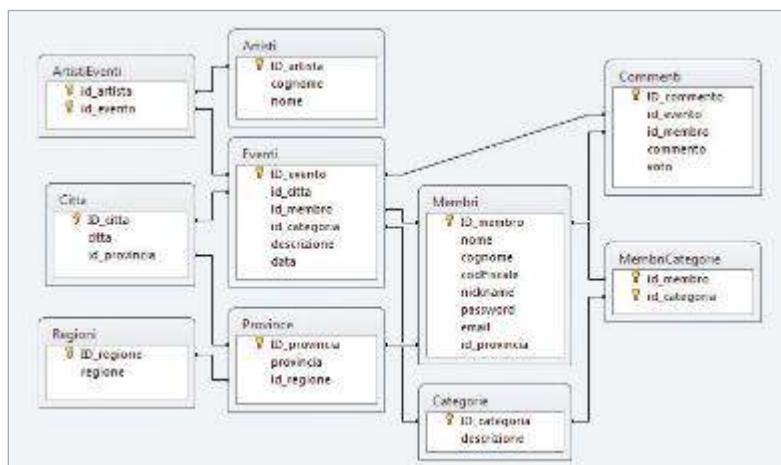
Tra gli attributi richiesti ai membri in fase di iscrizione risulta obbligatorio l'inserimento della provincia di appartenenza, in modo da poter poi inviare la newsletter mirata con gli eventi che si terranno nella sua zona.

Tabella Commenti

La tabella Commenti ha ID_commento come chiave primaria artificiale autoincrement:

Commenti

Nome campo	Tipo dati	Descrizione
ID_commento	Numerazione automat	pk
id_evento	Numerico	fk eventi
id_membro	Numerico	fk membro
commento	Testo	200 caratteri
voto	Numerico	Intero compreso tra 1 e 5
data	Data/ora	data inserimento commento

Tabelle ponte per le relazioni n, n

Le tabelle ponte che risolvono le relazioni (n, n) sono le seguenti:

ArtistiEventi

Nome campo	Tipo dati	Descrizione
id_artista	Numerico	fk artisti
id_evento	Numerico	fk eventi

Tra Artisti ed Eventi

MembriCategorie

Nome campo	Tipo dati	Descrizione
id_membro	Numerico	fk membri
id_categoria	Numerico	fk categorie

Tra Membri e Categorie

Tutte le tabelle sono consultabili nel file Eventi2015.mdb (cartella DATABASE del CD-ROM), del quale riportiamo lo schema E-R completo.

Relazioni della base di dati

Nel **punto 4** si richiede di riportare alcune relazioni della base di dati in cui siano presenti vincoli di integrità referenziale e/o vincoli di dominio: come parte del modello fisico, consideriamo significativa la tabella Commenti, nella quale sono presenti sia vincoli referenziali che vincoli di dominio.
Ne riportiamo la query di creazione, dove sono espresse le relazioni desiderate:

```
CREATE TABLE Commenti(
  ID_commento autoincrement Primary key,
  id_evento INT constraint x references Eventi(ID_evento),   /* vincolo di struttura */
  id_membro INT constraint y references Membri(ID_membro),   /* vincolo di struttura */
  commento VARCHAR(80) NOT NULL,                              /* vincolo di dominio   */
  voto INT CHECK (voto>0 AND voto<6),                         /* vincolo di dominio   */
  data TIMESTAMP
)
```

Interrogazioni

Nel **punto 5** sono richieste quattro interrogazioni di selezione.

Prima interrogazione

a) Elenco degli eventi già svolti, in ordine alfabetico di provincia.

```
$query1 = "
SELECT E.data, E.descrizione, C.citta, P.provincia
  FROM eventi AS E, Province AS P, Citta AS C
    WHERE E.id_citta=C.ID_citta AND C.id_provincia=P.ID_provincia AND E.data < $data1
      ORDER BY P.provincia";
```

Inserendo come $data1 il giorno “01-01-2018” si ottiene:

data	descrizione	citta	provincia
19/10/2017	mercatini di natale	Erba	Como
24/11/2017	incanto dei canestri	Como	Como
10/10/2017	processione serale	Cermenate	Como
12/10/2017	concerto serale	Cermenate	Como
16/11/2017	Festa del circolo	Novate Milanese	Milano
12/10/2017	Concerto in piazza	Milano	Milano
27/10/2017	Castagnata	Novate Milanese	Milano
13/12/2017	concerto di natale	Torino	Torino
13/09/2017	torneo di bocce	Torino	Torino

Seconda interrogazione

b) Elenco dei membri che non hanno mai inserito un commento.

```
SELECT nickname
  FROM Membri AS M
    WHERE M.ID_membro NOT IN
      (SELECT DISTINCT id_membro FROM Commenti)
        ORDER BY nickname
```

Una sua esecuzione produce il seguente risultato sui dati di prova presenti nel nostro archivio:

nickname
giorgio
simonetta

Terza interrogazione

c) Per ogni evento il voto medio ottenuto in ordine di categoria e titolo.

```
SELECT E.descrizione, AVG(C.voto) AS media
 FROM Eventi As E
   INNER JOIN Commenti As C ON C.id_evento=E.ID_Evento
     GROUP BY  E.ID_Evento, E.descrizione, E.id_categoria
       ORDER BY E.id_categoria, E.descrizione
```

Una sua esecuzione produce il seguente risultato sui dati di prova presenti nel nostro archivio:

descrizione	media valutazioni
Castagnata	4
Festa del circolo	3
Concerto in piazza	1.5
concerto serale	4

Quarta interrogazione

d) I dati dell'utente che ha registrato il maggior numero di eventi.

Questa richiesta, apparentemente non complessa, richiede invece una particolare attenzione.

Una soluzione semplice è quella che visualizza tutti i membri dopo aver contato per ciascuno di essi il numero di eventi inseriti, mostrando l'elenco ordinato in senso decrescente:

```
$query1 = "
  SELECT Count(E.id_membro) AS NrEventi, M.ID_membro, M.nickname, M.email
    FROM Eventi AS E INNER JOIN Membri AS M ON E.id_membro=M.id_membro
      GROUP BY M.ID_membro, M.nickname, M.email
        ORDER BY Count(E.id_membro) DESC
```

Ma la soluzione corretta, che visualizza solo chi ha inserito il maggior numero di eventi, è più complessa, in quanto richiede che preventivamente si individui il numero massimo di eventi inseriti da un utente e si utilizzi tale risultato come parametro nella selezione, come riportato di seguito:

```
SELECT *
FROM
 (SELECT Count(E.id_membro) AS NrEventi, M.ID_membro, M.nickname, M.email
  FROM Eventi AS E INNER JOIN Membri AS M ON E.id_membro = M.ID_membro
   GROUP BY M.ID_membro, M.nickname, M.email
 )
WHERE NrEventi =
 (SELECT MAX(NrEventi) AS MAGGIORE
  FROM
   (SELECT Count(E.id_membro) AS NrEventi, M.ID_membro
    FROM Eventi AS E INNER JOIN Membri AS M ON E.id_membro = M.ID_membro
     GROUP BY M.ID_membro
   )
 )
```

Una sua esecuzione produce il seguente risultato sui dati di prova presenti nel nostro archivio:

NrEventi	Membro	nickname	email
4	3	gragra	rossi@live.it
4	4	tom	tom@live.it

Sezione Web

Il **punto 6** richiede il progetto della pagina dell'interfaccia Web che permetta a un utente registrato di svolgere le operazioni specificate. L'utente può svolgere due attività:
– inserire un nuovo evento nell'archivio;
– postare un commento in merito a un evento a cui ha assistito.

Entrambe le richieste non presentano difficoltà: si tratta di disegnare il layout dei Form che permettono di effettuare le operazioni indicate.

Il **punto 7** richiede la codifica in un linguaggio a scelta di un segmento significativo che consenta l'interazione con la base di dati: scriveremo il codice php che visualizza l'elenco degli eventi presenti in archivio in base alla selezione di una provincia e di una categoria scelta dall'utente.

Essendo il codice di facile interpretazione, non lo commenteremo nei singoli dettagli.

La prima volta che viene richiamata, la pagina php permette all'utente di scegliere la categoria e la provincia nella quale si tiene l'evento:

```
<?php
// effettuo la connessione ADO ad access
include 'connettiEV.php';
// tecnica POSTBACK per realizzare il form nella stessa pagina
if (!$_POST)  // prima volta della sua esecuzione
{
  // predispongo il form
  $form="
  <table width='320' border='1'>
    <tr>
      <th width='160'><strong>Provincia</strong></th>
      <th width='160'><strong>Categoria</strong></th>
    </tr>
    </TABLE>";
  echo $form;
```

I dati vengono caricati in due ComboBox, "andandoli" a leggere dalle rispettive tabelle presenti nel database:
– nel primo ComboBox inseriamo i dati della tabella Province:

```
// primo combo box
$query1 = "SELECT ID_provincia, provincia FROM province ";
// apro la Connection
$cn->Open($sc);
// eseguo la query generando il primo Recordset
$rs->Open($query1, $cn);
echo "<FORM name='form1' ACTION='".$_SERVER['PHP_SELF']."' METHOD=POST><TABLE><TR>";
echo "<tr>";
echo " <th scope='col'><select name='provincia'>";
echo "<option value=0><'scegli la provincia'></option>";
// ciclo di lettura e stampa del recordset
while (!$rs->eof)
{                     // inserisce un dato nel combo box
  echo "<option value=".$rs->fields[0]->value."_".$rs->fields[1]->value.">".$rs->fields[1]->value."</option>";
  $rs->movenext();    // metodo movenext() per passare al record successivo
}
echo " </select></th>";
$rs->Close();
```

– nel secondo ComboBox inseriamo i dati della tabella Categorie:

```
// secondo combo box
$query2 = " SELECT ID_categoria, descrizione FROM Categorie ";
// eseguo la query generando il secondo Recordset
$rs->Open($query2, $cn);
echo "<th scope='col'><select name='categoria'>";
echo "<option value=0><'scegli la categoria'></option>";
while (!$rs->eof)
{
  echo "<option value=".$rs->fields[0]->value."_".$rs->fields[1]->value.">".$rs->fields[1]->value."</option>";
  $rs->movenext();
}
echo "  </select></th>";
echo " </tr>";
echo "</table>";
$rs->Close();
```

La seconda volta che è richiamata la pagina php si leggono i dati presenti nell'array $_POST[ ] per predisporre la query che andrà a prelevare dalla tabella Eventi le rappresentazioni che soddisfano i criteri di ricerca:

```
else
{
  // recupero le varabili selezionate
  $dato1 =$_POST['provincia'];
  $dato = explode("_",$dato1);  // crea un array con i due dati
  $dato1        = $dato[0];     // identificatore
  $provincia    = $dato[1];     // descrizione

  $dato2 =$_POST['categoria'];
  $dato = explode("_",$dato2);  // crea un array con i due dati
  $dato2       = $dato[0];      // identificatore
  $categoria   = $dato[1];      // descrizione

  // apro la connessione
  $cn->Open($sc);
  // predispongo l'interrogazione
  $query3="
   SELECT ID_evento AS campo1, citta AS campo2, descrizione AS campo3, data AS campo4
    FROM eventi as E, province AS P, citta AS C
      WHERE P.ID_provincia=".$dato1." AND id_categoria=".$dato2."
        AND E.id_citta = C.id_citta AND P.ID_provincia=C.id_provincia
          ORDER BY data ASC
  ";
  echo $query3;
```

I risultati della query sono quindi visualizzati sullo schermo mediante un ciclo che analizza il recordset ottenuto:

```
// stampa tabella risultati con riga di intestazione
if ($rs->eof) // trovato accetta credenziali
   echo '<br>Nessun evento disponibile per le scelte effettuate';
else
  // ciclo di lettura e stampa del recordset
  while (!$rs->eof)
  {
    echo "<TR>";
    // stampa del singolo campo in ogni cella
    echo'<TD> <center>'.$rs->fields['campo1']->value."</TD>";
    echo'<TD>'.$rs->fields['campo2']->value."</TD>";
    echo'<TD>'.$rs->fields['campo3']->value."</TD>";
    echo'<TD> <center>'.$rs->fields['campo4']->value."</TD>";
    echo'</TR>';
    // metodo movenext() per passare al record successivo
    $rs->movenext();
  }
```

Selezionando la provincia di Milano e come categoria il balletto otteniamo il seguente risultato:

Provincia	Categoria
Milano	balletto

ID_evento	citta	descrizione	data
12	Novate Milanese	lo schiaccianoci	22/12/2017

SECONDA PARTE della traccia

Riportiamo la soluzione ai quattro quesiti richiesti come **SECONDA PARTE** della prova d'esame.

Integrazione del progetto con l'aggiunta di inserzioni pubblicitarie

Il **primo quesito** richiede di integrare quanto progettato nella **PRIMA PARTE** aggiungendo la possibilità di inserire e gestire inserzioni pubblicitarie catalogate e composte da un testo e un link, che vengono visualizzate in modo mirato a membri che hanno indicato una data categoria come preferenza, oppure che hanno visitato eventi di tale categoria.

Aggiungiamo al database un'entità Inserzione in relazione alla entità Categoria del tipo (1, n): ogni inserzione è cioè specifica per una categoria di evento.

Il tracciato record della tabella Inserzioni è il seguente:

Inserzioni

Nome campo	Tipo dati	Descrizione
ID_Inserzione	Numerazione automat	pf
data	Data/ora	data inserimento
scadenza	Data/ora	data scadenza
descrizione	Testo	255 caratteri
link	Testo	80 caratteri
id_categoria	Numerico	fk tabella categoria

È inoltre necessario inserire un controllo all'interno del programma di gestione che aggiorna la tabella ponte MembriCategorie ogniqualvolta un membro effettui una ricerca di un evento in base a una categoria e ne visualizzi le informazioni, in modo da completare dinamicamente le preferenze affinché l'inserzione pubblicitaria possa avere il maggior successo possibile.

Progetto del layout di una pagina Web

Per rispondere al **secondo quesito**, che richiede di garantire un aspetto grafico comune per tutte le parti del sito, si rimanda alla trattazione delle pagine realizzate mediante CSS, oppure all'utilizzo di template tramite CMS.

Normalizzazione di una tabella

La tabella proposta nel **terzo quesito** ha il seguente schema logico:

CorsiIscrizioni (Cognome, Nome, Telefono, Livello, Tutor, Tel-tutor, AnticipoVersato)

Non è sicuramente in 1 FN, dato che in essa non è definita alcuna chiave primaria: tra le varie possibilità, la chiave formata dai due attributi Telefono e Tel-tutor è una chiave candidata, oppure si può prevedere l'aggiunta di una chiave artificiale, e noi opteremo per la prima scelta.
Una volta definita la chiave primaria, dato che gli attributi sono atomici, la tabella è in 1 FN ma non in 2 FN, perché esistono dipendenze funzionali parziali di alcuni attributi rispetto alla chiave scelta:
– Cognome e Nome dipendono soltanto dall'attributo Telefono;
– Tutor dipende soltanto dall'attributo Tel-tutor.

Per eliminare la ridondanza dei dati è necessario scorporare i dati con dipendenza parziale, per cui la tabella iniziale viene scomposta nelle seguenti tre tabelle:

AnticipiVersati (Telefono(pk), Tel-tutor(pk), Livello, AnticipoVersato)
Utenti (Telefono (pk), Cognome, Nome)
Tutor (Tel-tutor (pk), Tutor)

Gli attributi Livello e AnticipoVersato dipendono dall'intera chiave scelta per la tabella AnticipiVersati e quindi la struttura così ottenuta soddisfa le richieste di normalizzazione.

Domanda teorica sulla cardinalità delle associazioni

La risposta al **quarto quesito** teorico viene descritta nella **UDA 1 Lezione 5**, alla quale si rimanda.